



## Multi-access Edge Computing (MEC); Edge Platform Application Enablement

**Disclaimer:** This DRAFT is a working document of ETSI ISG MEC. It is provided for information only and is still under development within ETSI ISG MEC. ETSI and its Members accept no liability for any further use/implementation of this Specification.

Non-published MEC drafts stored in the ["Open Area"](#) are working documents, these may be updated, replaced, or removed at any time

**Do not use as reference material.**

*Disclaimer*

**Do not cite this document other than as "work in progress".**

The present document has been produced and approved by the Multi-access Edge Computing (MEC) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG. It does not necessarily represent the views of the entire ETSI membership.

Approved and published Specifications and reports for implementation of the MEC system shall be obtained via the ETSI Standards search page at:  
<http://www.etsi.org/standards-search>

---

**Reference**

RGS/MEC-0011v311Plat.App.Enab

---

**Keywords**

API, MEC

**ETSI**

650 Route des Lucioles  
F-06921 Sophia Antipolis Cedex - FRANCE

---

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B  
Association à but non lucratif enregistrée à la  
Sous-Préfecture de Grasse (06) N° w061004871

---

**Important notice**

The present document can be downloaded from:

<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:

<https://portal.etsi.org/People/CommitteeSupportStaff.aspx>

---

**Notice of disclaimer & limitation of liability**

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

---

**Copyright Notification**

Reproduction is only permitted for the purpose of standardization work undertaken within ETSI.

The copyright and the foregoing restrictions extend to reproduction in all media.

© ETSI 2021.  
All rights reserved.

# Contents

Intellectual Property Rights .....	8
Foreword.....	8
Modal verbs terminology.....	8
1 Scope .....	9
2 References .....	9
2.1 Normative references .....	9
2.2 Informative references.....	10
3 Definition of terms, symbols and abbreviations.....	11
3.1 Terms.....	11
3.2 Symbols.....	11
3.3 Abbreviations .....	11
4 Overview .....	12
5 Description of the services (informative).....	12
5.1 Introduction .....	12
5.2 Sequence diagrams .....	13
5.2.1 General.....	13
5.2.2 MEC application start-up.....	13
5.2.3 MEC application graceful termination/stop.....	16
5.2.4 Service availability update and new service registration .....	16
5.2.5 Service availability query .....	18
5.2.6 Managing subscription to event notifications .....	19
5.2.6.1 Introduction .....	19
5.2.6.2 Subscribing to event notifications .....	19
5.2.6.3 Unsubscribing from event notifications .....	20
5.2.7 Traffic rule activation/deactivation/update.....	20
5.2.8 DNS rule activation/deactivation.....	20
5.2.9 Transport information query.....	21
5.2.10 Time of Day (ToD).....	21
5.2.10.1 Introduction.....	21
5.2.10.2 Get platform time.....	22
5.2.10.3 Timing capabilities query flow .....	22
5.2.11 Service deregistration .....	23
5.2.12 Service heartbeat.....	23
5.2.13 MEC application registration.....	24
5.2.13.1 Introduction.....	24
5.2.13.2 Application registration.....	24
6 Common data types .....	25
6.1 Introduction .....	25
6.2 Resource data types .....	25
6.2.1 Introduction.....	25
6.2.2 Type: SubscriptionLinkList .....	25
6.3 Referenced structured data types.....	25
6.3.1 Introduction.....	25
6.3.2 Type: LinkType .....	25
7 MEC application support API.....	26
7.1 Data model .....	26
7.1.1 Introduction.....	26
7.1.2 Resource data types .....	26
7.1.2.1 Introduction.....	26
7.1.2.2 Type: TrafficRule.....	26
7.1.2.3 Type: DnsRule .....	27
7.1.2.4 Type: TimingCaps.....	27

7.1.2.5	Type: CurrentTime.....	28
7.1.2.6	Type: AppInfo.....	29
7.1.3	Subscription data types .....	30
7.1.3.1	Introduction.....	30
7.1.3.2	Type: AppTerminationNotificationSubscription.....	30
7.1.4	Notification data types .....	30
7.1.4.1	Introduction.....	30
7.1.4.2	Type: AppTerminationNotification.....	30
7.1.4.3	Type: AppTerminationConfirmation .....	31
7.1.4.4	Type: AppReadyConfirmation.....	31
7.1.5	Referenced structured data types .....	31
7.1.5.1	Introduction.....	31
7.1.5.2	Type: TrafficFilter.....	31
7.1.5.3	Type: DestinationInterface.....	32
7.1.5.4	Type: TunnelInfo .....	32
7.1.6	Referenced simple data types and enumerations .....	33
7.2	API definition.....	33
7.2.1	Introduction.....	33
7.2.2	Global definitions and resource structure .....	33
7.2.3	Resource: all mecAppSupportSubscription .....	35
7.2.3.1	Description .....	35
7.2.3.2	Resource definition .....	35
7.2.3.3	Resource methods .....	35
7.2.3.3.1	GET .....	35
7.2.3.3.2	PUT .....	36
7.2.3.3.3	PATCH.....	36
7.2.3.3.4	POST .....	36
7.2.3.3.5	DELETE.....	37
7.2.4	Resource: individual mecAppSupportSubscription .....	37
7.2.4.1	Description .....	37
7.2.4.2	Resource definition .....	37
7.2.4.3	Resource methods .....	37
7.2.4.3.1	GET .....	37
7.2.4.3.2	PUT .....	38
7.2.4.3.3	PATCH.....	38
7.2.4.3.4	POST .....	38
7.2.4.3.5	DELETE.....	38
7.2.5	Resource: mecTimingCaps .....	39
7.2.5.1	Description.....	39
7.2.5.2	Resource definition .....	39
7.2.5.3	Resource methods .....	40
7.2.5.3.1	GET .....	40
7.2.5.3.2	PUT .....	40
7.2.5.3.3	PATCH.....	40
7.2.5.3.4	POST.....	41
7.2.5.3.5	DELETE.....	41
7.2.6	Resource: mecCurrentTime .....	41
7.2.6.1	Description.....	41
7.2.6.2	Resource definition .....	41
7.2.6.3	Resource methods .....	41
7.2.6.3.1	GET .....	41
7.2.6.3.2	PUT .....	42
7.2.6.3.3	PATCH.....	42
7.2.6.3.4	POST.....	42
7.2.6.3.5	DELETE.....	42
7.2.7	Resource: all mecTrafficRule .....	42
7.2.7.1	Description.....	42
7.2.7.2	Resource definition .....	42
7.2.7.3	Resource methods .....	43
7.2.7.3.1	GET .....	43
7.2.7.3.2	PUT .....	43
7.2.7.3.3	PATCH.....	43

7.2.7.3.4	POST .....	43
7.2.7.3.5	DELETE .....	43
7.2.8	Resource: individual mecTrafficRule .....	43
7.2.8.1	Description .....	43
7.2.8.2	Resource definition .....	44
7.2.8.3	Resource methods .....	44
7.2.8.3.1	GET .....	44
7.2.8.3.2	PUT .....	44
7.2.8.3.3	PATCH .....	45
7.2.8.3.4	POST .....	46
7.2.8.3.5	DELETE .....	46
7.2.9	Resource: all mecDnsRule .....	46
7.2.9.1	Description .....	46
7.2.9.2	Resource definition .....	46
7.2.9.3	Resource methods .....	46
7.2.9.3.1	GET .....	46
7.2.9.3.2	PUT .....	47
7.2.9.3.3	PATCH .....	47
7.2.9.3.4	POST .....	47
7.2.9.3.5	DELETE .....	47
7.2.10	Resource: individual mecDnsRule .....	47
7.2.10.1	Description .....	47
7.2.10.2	Resource definition .....	47
7.2.10.3	Resource methods .....	48
7.2.10.3.1	GET .....	48
7.2.10.3.2	PUT .....	48
7.2.10.3.3	PATCH .....	49
7.2.10.3.4	POST .....	49
7.2.10.3.5	DELETE .....	49
7.2.11	Resource: confirm termination task .....	49
7.2.11.1	Description .....	49
7.2.11.2	Resource definition .....	49
7.2.11.3	Resource methods .....	50
7.2.11.3.1	GET .....	50
7.2.11.3.2	PUT .....	50
7.2.11.3.3	PATCH .....	50
7.2.11.3.4	POST .....	50
7.2.11.3.5	DELETE .....	51
7.2.12	Resource: confirm ready task .....	51
7.2.12.1	Description .....	51
7.2.12.2	Resource definition .....	51
7.2.12.3	Resource methods .....	52
7.2.12.3.1	GET .....	52
7.2.12.3.2	PUT .....	52
7.2.12.3.3	PATCH .....	52
7.2.12.3.4	POST .....	52
7.2.12.3.5	DELETE .....	53
8	MEC service management API .....	53
8.1	Data model .....	53
8.1.1	Introduction .....	53
8.1.2	Resource data types .....	53
8.1.2.1	Introduction .....	53
8.1.2.2	Type: ServiceInfo .....	53
8.1.2.3	Type: TransportInfo .....	54
8.1.2.4	Type: ServiceLivenessInfo .....	55
8.1.2.5	Type: ServiceLivenessUpdate .....	55
8.1.3	Subscription data types .....	55
8.1.3.1	Introduction .....	55
8.1.3.2	Type: SerAvailabilityNotificationSubscription .....	56
8.1.4	Notification data types .....	56
8.1.4.1	Introduction .....	56

8.1.4.2	Type: ServiceAvailabilityNotification .....	56
8.1.5	Referenced structured data types .....	57
8.1.5.1	Introduction .....	57
8.1.5.2	Type: CategoryRef .....	57
8.1.5.3	Type: EndPointInfo .....	57
8.1.5.4	Type: SecurityInfo .....	58
8.1.6	Referenced simple data types and enumerations .....	59
8.1.6.1	Introduction .....	59
8.1.6.2	Simple data types .....	59
8.1.6.3	Enumeration: SerializerType .....	59
8.1.6.4	Enumeration: TransportType .....	59
8.1.6.5	Enumeration: LocalityType .....	59
8.1.6.6	Enumeration: ServiceState .....	60
8.2	API definition .....	60
8.2.1	Introduction .....	60
8.2.2	Global definitions and resource structure .....	60
8.2.3	Resource: a list of mecService .....	62
8.2.3.1	Description .....	62
8.2.3.2	Resource definition .....	62
8.2.3.3	Resource methods .....	62
8.2.3.3.1	GET .....	62
8.2.3.3.2	PUT .....	63
8.2.3.3.3	PATCH .....	63
8.2.3.3.4	POST .....	64
8.2.3.3.5	DELETE .....	64
8.2.4	Resource: individual mecService .....	64
8.2.4.1	Description .....	64
8.2.4.2	Resource definition .....	64
8.2.4.3	Resource methods .....	64
8.2.4.3.1	GET .....	64
8.2.4.3.2	PUT .....	65
8.2.4.3.3	PATCH .....	65
8.2.4.3.4	POST .....	65
8.2.4.3.5	DELETE .....	65
8.2.5	Resource: a list of mecTransport .....	65
8.2.5.1	Description .....	65
8.2.5.2	Resource definition .....	65
8.2.5.3	Resource methods .....	66
8.2.5.3.1	GET .....	66
8.2.5.3.2	PUT .....	66
8.2.5.3.3	PATCH .....	67
8.2.5.3.4	POST .....	67
8.2.5.3.5	DELETE .....	67
8.2.6	Resource: a list of mecService of an application instance .....	67
8.2.6.1	Description .....	67
8.2.6.2	Resource definition .....	67
8.2.6.3	Resource methods .....	67
8.2.6.3.1	GET .....	67
8.2.6.3.2	PUT .....	69
8.2.6.3.3	PATCH .....	69
8.2.6.3.4	POST .....	69
8.2.6.3.5	DELETE .....	70
8.2.7	Resource: individual mecService of an application instance .....	70
8.2.7.1	Description .....	70
8.2.7.2	Resource definition .....	70
8.2.7.3	Resource methods .....	70
8.2.7.3.1	GET .....	70
8.2.7.3.2	PUT .....	71
8.2.7.3.3	PATCH .....	72
8.2.7.3.4	POST .....	72
8.2.7.3.5	DELETE .....	72
8.2.8	Resource: all mecSrvMgmtSubscription .....	73

8.2.8.1	Description .....	73
8.2.8.2	Resource definition .....	73
8.2.8.3	Resource methods .....	73
8.2.8.3.1	GET .....	73
8.2.8.3.2	PUT .....	74
8.2.8.3.3	PATCH .....	74
8.2.8.3.4	POST .....	74
8.2.8.3.5	DELETE .....	75
8.2.9	Resource: individual mecSrvMgmtSubscription .....	75
8.2.9.1	Description .....	75
8.2.9.2	Resource definition .....	75
8.2.9.3	Resource methods .....	76
8.2.9.3.1	GET .....	76
8.2.9.3.2	PUT .....	76
8.2.9.3.3	PATCH .....	76
8.2.9.3.4	POST .....	76
8.2.9.3.5	DELETE .....	76
8.2.10	Resource: individual mecServiceLiveness .....	77
8.2.10.1	Description .....	77
8.2.10.2	Resource definition .....	77
8.2.10.3	Resource methods .....	78
8.2.10.3.1	GET .....	78
8.2.10.3.2	PUT .....	78
8.2.10.3.3	PATCH .....	78
8.2.10.3.4	POST .....	80
8.2.10.3.5	DELETE .....	80
<b>Annex A (informative): Complementary material for API utilization .....</b>		<b>81</b>
<b>Annex B (informative): Mapping MEC service management API to 3GPP CAPIF APIs .....</b>		<b>82</b>
B.0	Definitions (ETSI TS 123 222) .....	82
B.1	Introduction .....	82
B.2	Mapping MEC service management API to CAPIF APIs .....	83
B.2.1	Overview .....	83
B.2.2	Mapping of the resource structures .....	83
B.2.3	Data models for service API discovery and publication .....	84
B.2.3.1	Data model for services .....	84
B.2.3.2	Data model for service API announcement/notification .....	85
<b>Annex C (informative): Analysis of EASProfile .....</b>		<b>87</b>
History .....		89

---

# Intellectual Property Rights

## Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<https://ipr.etsi.org/>).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

## Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

**DECT™**, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

---

# Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Multi-access Edge Computing (MEC).

---

# Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the [ETSI Drafting Rules](#) (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.



---

# 1 Scope

The present document focuses on the functionalities enabled via the Mp1 reference point between MEC applications and MEC platform, which allows these applications to interact with the MEC system. Service related functionality includes registration/deregistration, discovery and event notifications. Other functionality includes application availability, traffic rules, DNS and time of day. It describes the information flows, required information, and specifies the necessary operations, data models and API definitions.

---

## 2 References

### 2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at <https://docbox.etsi.org/Reference>.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

- [1] ETSI GS MEC 001: "Multi-access Edge Computing (MEC) Terminology".
- [2] ETSI GS MEC 002: "Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements".
- [3] ETSI GS MEC 003: "Multi-access Edge Computing (MEC); Framework and Reference Architecture".
- [4] ETSI GS MEC 010-2: "Multi-access Edge Computing (MEC); MEC Management; Part 2: Application lifecycle, rules and requirements management".
- [5] ETSI GS MEC 009: "Multi-access Edge Computing (MEC); General principles, patterns and common aspects of MEC Service APIs".
- [6] Void.
- [7] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

NOTE: Available at <https://tools.ietf.org/html/rfc5246>.

- [8] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

NOTE: Available at <https://tools.ietf.org/html/rfc3986>.

- [9] IETF RFC 7159: "The JavaScript Object Notation (JSON) Data Interchange Format".

NOTE: Available at <https://tools.ietf.org/html/rfc7159>.

- [10] W3C Recommendation (16 August 2006): "Extensible Markup Language (XML) 1.1 (Second Edition)", edited in place 29 September 2006.

NOTE: Available at <https://www.w3.org/TR/xml11/>.

- [11] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1) Message Syntax and Routing".

NOTE: Available at <https://tools.ietf.org/html/rfc7230>.

- [12] IETF RFC 6455: "The WebSocket Protocol".  
NOTE: Available at <https://tools.ietf.org/html/rfc6455>.
- [13] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".  
NOTE: Available at <https://tools.ietf.org/html/rfc6749>.
- [14] IETF RFC 6750: "The OAuth 2.0 Authorization Framework: Bearer Token Usage".  
NOTE: Available at <https://tools.ietf.org/html/rfc6750>.
- [15] ETSI GS NFV-IFA 007: "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification".
- [16] IETF RFC 5789: "PATCH Method for HTTP".  
NOTE: Available at <https://tools.ietf.org/html/rfc5789>.
- [17] IETF RFC 7386: "JSON Merge Patch".  
NOTE: Available at <https://tools.ietf.org/html/rfc7386>.
- [18] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".  
NOTE: Available at <https://tools.ietf.org/html/rfc8446>.
- [19] 3GPP TS 29.558: "Enabling Edge Applications; Application Programming Interface (API) specification; Stage 3 (Release 17)".
- [20] ETSI GS MEC 016: "Multi-access Edge Computing (MEC); Device application interface".
- [21] ETSI TS 123 558: "Architecture for enabling Edge Applications; (Release 17)" (3GPP TS 23.558).

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] IETF RFC 5905: "Network Time Protocol Version 4: Protocol and Algorithms Specification".
- [i.2] IEEE 1588-2019™: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems".
- [i.3] Protocol buffers, version 3.  
NOTE: Available at <https://developers.google.com/protocol-buffers/docs/proto3>.
- [i.4] OASIS Standard: "MQTT Version 3.1.1", 29 October 2014.  
NOTE: Available at <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>.
- [i.5] gRPC™.
- NOTE: Available at <http://www.grpc.io/>.
- [i.6] OpenAPI™ Specification.  
NOTE: Available at <https://github.com/OAI/OpenAPI-Specification>.

[i.7] IETF RFC 4122: "A Universally Unique Identifier (UUID) URN Namespace".

NOTE: Available at <https://tools.ietf.org/html/rfc4122>.

[i.8] ETSI TS 123 222: "LTE; 5G; Common API Framework for 3GPP Northbound APIs (3GPP TS 23.222 Release 16)".

[i.9] ETSI TS 129 222: "5G; LTE; Common API Framework for 3GPP Northbound APIs (3GPP TS 29.222 Release 16)".

## 3 Definition of terms, symbols and abbreviations

### 3.1 Terms

For the purposes of the present document, the terms given in ETSI GS MEC 001 [1] apply.

### 3.2 Symbols

Void.

### 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS MEC 001 [1] and the following apply:

5GS	5G System
API	Application Programming Interface
CAPIF	Common API Framework
CCF	CAPIF Core Function
DSCP	Differentiated Services Code Point
E-UTRA	Evolved Universal Terrestrial Radio Access
EPS	Evolved Packet System
FQDN	Fully Qualified Domain Name
GRE	Generic Routing Encapsulation
GTP	GPRS Tunnelling Protocol
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
MAC	Media Access Control
MQTT	Message Queue Telemetry Transport
NFVI	Network Functions Virtualisation Infrastructure
NTP	Network Time Protocol
NR	New Radio
PoP	Point of Presence
PTP	Precision Time Protocol
QCI	Quality Class Indicator
REST	Representational State Transfer
RFC	Request For Comments
RNI	Radio Network Information
RPC	Remote Procedure Call
TC	Traffic Class
TLS	Transport Layer Security
ToD	Time of Day
URI	Uniform Resource Indicator
UTC	Coordinated Universal Time
VNF	Virtualised Network Function
XML	eXtensible Markup Language

---

## 4 Overview

The present document specifies two MEC Platform Application Enablement APIs that support the requirements defined for Multi-access Edge Computing in ETSI GS MEC 002 [2], namely the MEC application support API and the MEC service management API.

Clause 5 introduces the functionalities enabled via the Mp1 reference point between MEC applications and MEC platform. It provides the high level information flows and describes the necessary operations.

The common data types are defined in clause 6, while the corresponding data models and API definitions are specified in clause 7 for the MEC application support API and clause 8 for the MEC service management API.

---

## 5 Description of the services (informative)

### 5.1 Introduction

The MEC platform, as defined in ETSI GS MEC 003 [3], offers an environment where MEC applications may discover, advertise, consume and offer MEC services. Upon receipt of update, activation or deactivation of traffic rules from the MEC platform manager, applications or services, the MEC platform instructs the data plane accordingly. The MEC platform also receives DNS records from the MEC platform manager and uses them to configure a DNS proxy/server.

Via Mp1 reference point between the MEC platform and the MEC applications, as defined in ETSI GS MEC 003 [3], the basic functions are enabled, such as:

- MEC service assistance:
  - authentication and authorization of producing and consuming MEC services;
  - a means for service producing MEC applications to register/deregister towards the MEC platform the MEC services they provide, and to update the MEC platform about changes of the MEC service availability;
  - a means to notify the changes of the MEC service availability to the relevant MEC application;
  - discovery of available MEC services;
- MEC application assistance:
  - MEC application start-up procedure;
  - MEC application graceful termination/stop;
- traffic routing:
  - traffic rules update, activation and deactivation;
- DNS rules:
  - DNS rules activation and deactivation;
- timing:
  - providing access to time of day information;
- transport information:
  - providing information about available transports.

These functions are grouped into those considered to provide MEC application support (i.e. application specific traffic routing, DNS rules and timing, as well as graceful termination/stop) and those that provide MEC service management (i.e. MEC service assistance and associated service transport information).

## 5.2 Sequence diagrams

### 5.2.1 General

Clauses 5.2.2 to 5.2.10 describe how MEC applications and/or MEC services may be supported by the MEC platform via Mp1 reference point. The related sequence diagrams are presented.

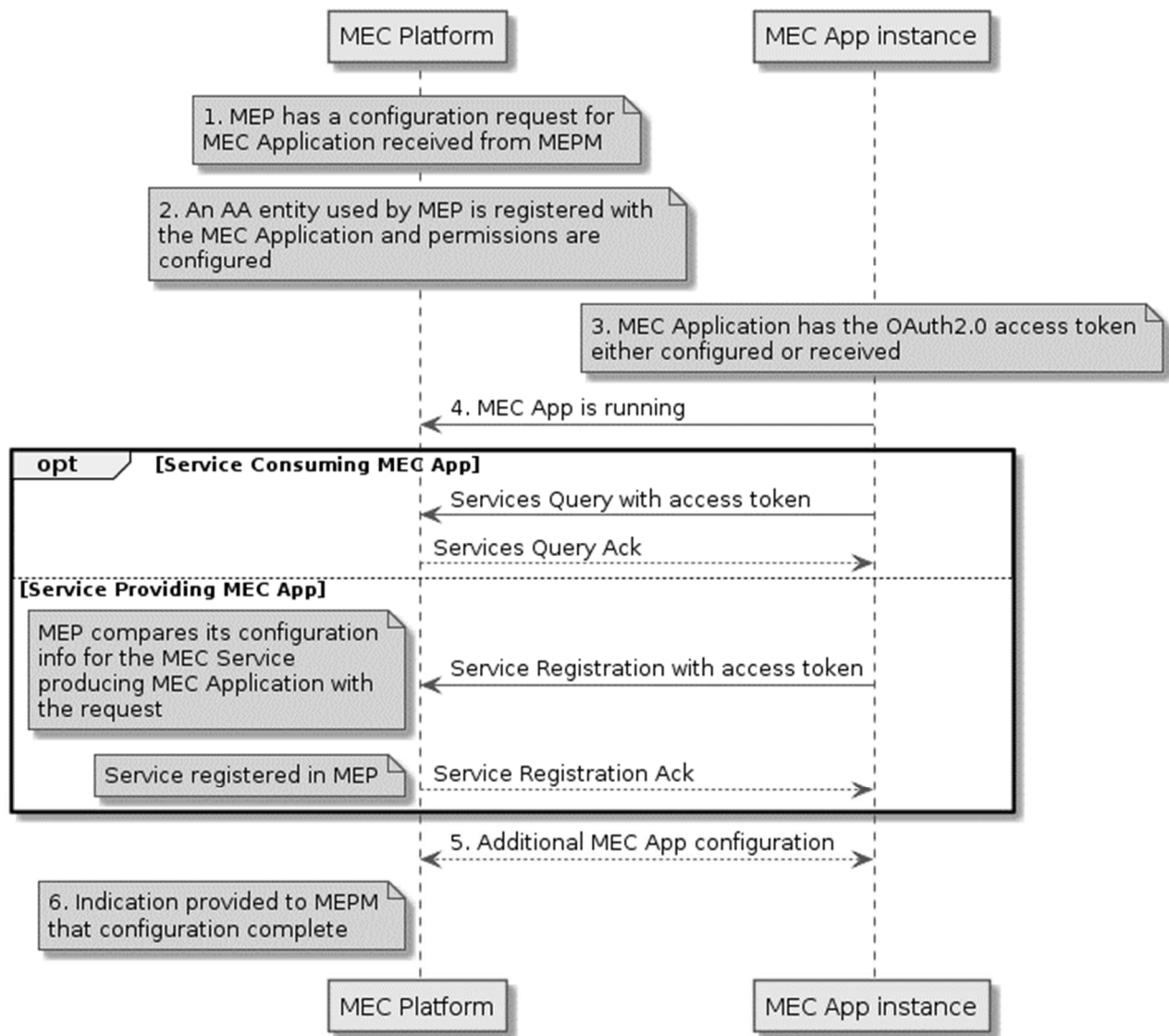
### 5.2.2 MEC application start-up

Figure 5.2.2-1 shows three alternative messages that a MEC application can use to communicate with a MEC platform during the start-up phase of the application instantiation process, steps 5 to 7 in clause 5.3.1 of ETSI GS MEC 010-2 [4].

In this flow, the MEC platform can verify the authenticity of the MEC application with the aid of an AA entity that contains the registration related information about the MEC application in question. For actual authentication, the MEC application uses access token based on OAuth2.0.

MEC platform also has possibility to verify the correctness of the service registration or services query of the MEC application, as it is assumed that MEC platform has received the valid configuration for service consuming and service producing MEC applications. The related information about this MEC application instance (including the required and the optional services, the services to be offered by this application instance and the associated transport dependency, the traffic rules and DNS rules associated with this application instance, etc.) can be compared to those included in the service registration or services query messages, which can be used to determine whether to accept or reject the request.

**Draft**



**Figure 5.2.2-1: Flow of MEC application start up**

MEC application start up procedure, following the MEC application instantiation procedure (as defined in ETSI GS MEC 010-2 [4]), consists of the following steps:

- 1) MEC platform has received a configuration request from MEC Platform Manager. The configuration request contains detailed information about the parameters related to the MEC application, including the required and the optional services, the services to be offered by this application instance and the associated transport dependency, the traffic rules and DNS rules associated with this application instance, etc.
- 2) An AA entity associated with the MEC platform has been configured with the MEC application related identity and permissions.
- 3) MEC application that intends to communicate with MEC platform has the OAuth2.0 access token either received or configured.

- 4) The MEC application that intends to communicate with MEC sends a "MEC App is running" message towards the MEC platform to confirm that the instantiation and the start-up phase have been successfully completed. If the application receives an error response with 409 status code from the platform, it should retry sending such message for a configurable period of time. This behaviour increases robustness to race-conditions in the instantiation process, in particular when the MEC platform has not yet received the configuration request from MEC Platform Manager, or the MEC platform is still processing the configuration request.

Depending on the nature of the MEC application and its intended use of MEC services, after the successful response received the MEC application may apply one or both of the following options:

a) Option 1:

Send services query to the MEC platform (MEC Application that consumes MEC Services). The services query request contains the access token.

b) Option 2:

Send a service registration request to the MEC platform (MEC application that provides MEC service(s)). The service registration request contains the access token. The MEC platform then compares the configuration it has for the service producing MEC application with the request, and if acceptable, registers the MEC service and returns a service registration acknowledgement.

NOTE 1: It is out of scope how a MEC application instance discovers a MEC platform. In practise, this may be statically configured or dynamically discovered via e.g. DNS.

- 5) If applicable, additional configuration on the MEC services may be performed between the MEC platform and MEC application.

The MEC system may also pre-configure (not through Mp1) the MEC application instance with necessary parameters, for example:

- the information needed to access the required services;
- the availability of the optional services;
- the information needed to access the available optional services.

The additional procedures via Mp1 that are related to this step include, when required, "Traffic rule activation/deactivation/update" as defined in clause 5.2.7, and "DNS rule activation/deactivation" as defined in clause 5.2.8. And the MEC application instance may update the MEC platform with the information about the available produced MEC services as defined in clause 5.2.4.

- 6) MEC platform sends an indication to MEC Platform Manager once the configuration is complete. This message is not further specified in the present document.

NOTE 2: The options 4a) and 4b) present different messages that can be sent by a MEC application. As MEC application can both consume and provide MEC service(s), it is possible that such MEC application performs both services query and service registration steps, in any order.

### 5.2.3 MEC application graceful termination/stop

Figure 5.2.3-1 shows a flow for MEC application instance graceful termination/stop (as defined in ETSI GS MEC 010-2 [4]). After the MEC platform receives a request to terminate or stop a MEC application instance the MEC platform notifies the MEC application instance that it will be terminated or stopped soon if graceful termination/stop is required. In the notification, the MEC platform indicates to the MEC application instance the time interval for the application to perform application-specific termination/stop actions. The time interval is set according to the graceful termination/stop timeout value in the received request to terminate or stop. When this timer expires, the MEC platform continues the termination flow of the MEC application instance or stop MEC application instance flow by, e.g. deactivating the traffic rules and DNS rules, removing the MEC application instance from the list of instances to be notified about service availability, removing the services provided by the MEC application instance from the service registry, sending service availability notification to the MEC applications that consumes the services produced by the terminating/stopping MEC application instance, etc.

The MEC application instance has the option to, before the timer expires, inform the MEC platform that it is ready to be terminated/stopped after it has finished any application level related actions. Upon receipt of this information, the MEC platform continues the flow to terminate or stop the MEC application instance. The service producing MEC application instance should also deregister its produced MEC service(s) towards the MEC platform before the timer expires. Upon receipt of the request, the MEC platform deregisters the MEC service(s).

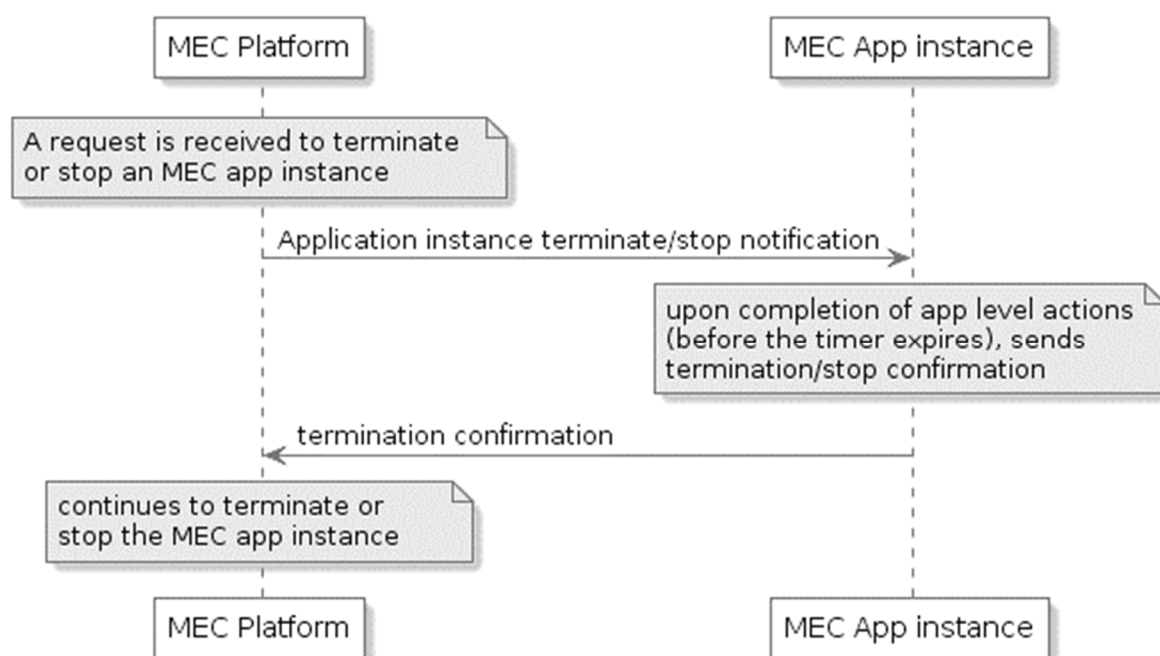


Figure 5.2.3-1: Example flow of MEC application instance graceful termination/stop

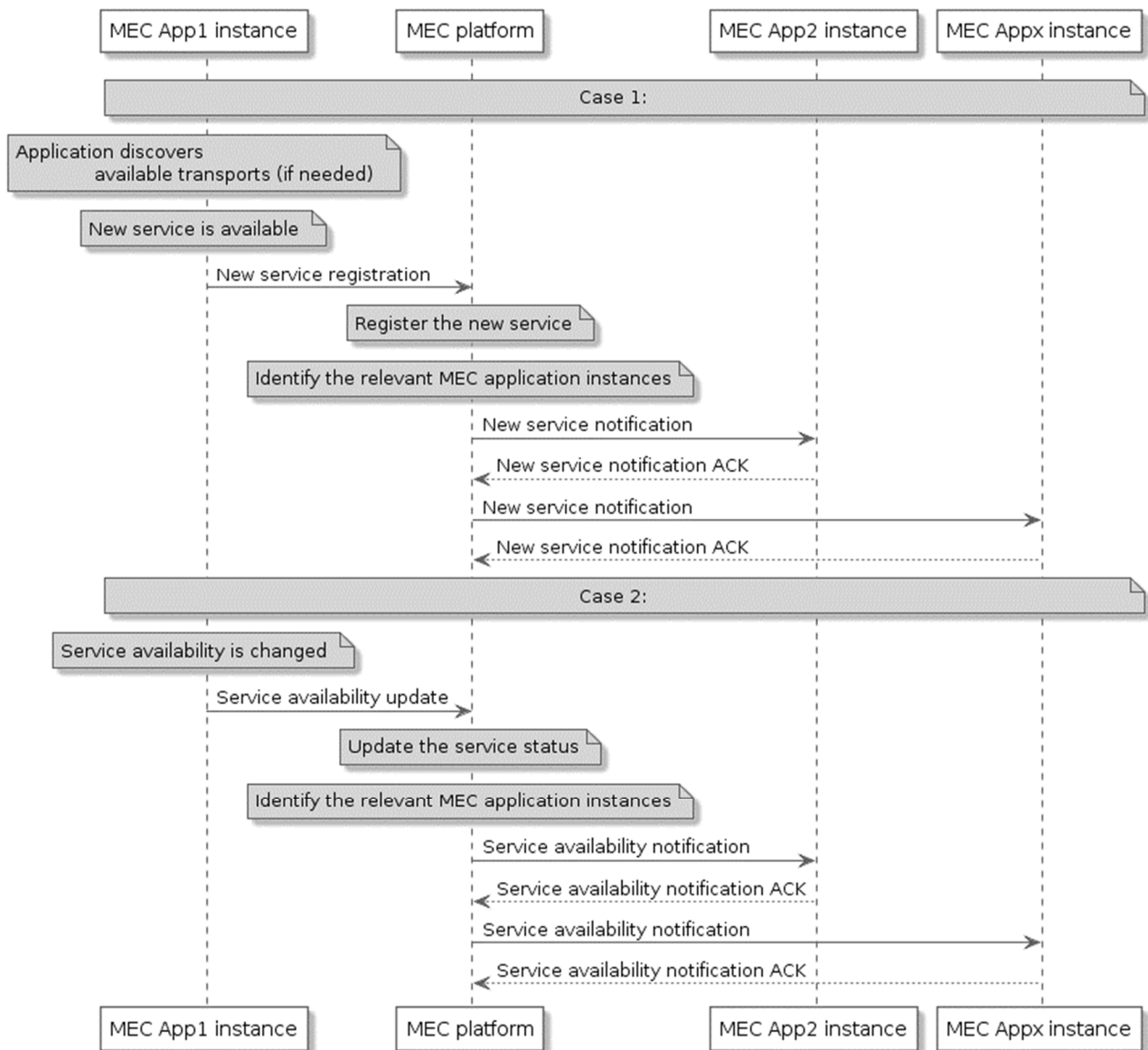
### 5.2.4 Service availability update and new service registration

When a MEC service is registered by the service producing MEC application, the authorized relevant applications (e.g. the applications that indicate the service as "optional" or "required") will be notified about the newly available service. Moreover, the authorized relevant applications will also be notified about the service availability changes of that service.

Figure 5.2.4-1 shows two cases. In the 1<sup>st</sup> case a MEC application instance informs the MEC platform that the service(s) provided by this application instance become available for the first time (service registration); and then the MEC platform notifies the authorized relevant application instances (e.g. the applications that indicate the service(s) as "optional" or "required") about the newly available service(s). As part of service registration, the relevant information about the service is provided to the platform, and the service is bound to a transport that is either provided by the MEC platform, or by the application itself.

In the 2<sup>nd</sup> case the service producing MEC application instance updates the MEC platform about the status change of the produced MEC services; and the MEC platform notify the authorized relevant application instances about the service availability changes.





**Figure 5.2.4-1: Flow of new service registration and service availability update**

In the 1<sup>st</sup> case the new service registration procedure consists of the following steps:

- 1) If the application intends to use a transport that is provided by the MEC platform, it discovers the available transports first, and selects one (or more) for use with the new service.
- 2) After a new MEC service becomes available, the service producing MEC application instance sends new service registration message to the MEC platform.
- 3) MEC platform registers the new service with the service registry. This step is not to be further specified.
- 4) MEC platform then identifies the relevant MEC application instance for this update (e.g. the applications that indicate the service as "optional" or "required"), and sends new service notifications to the relevant application instances. When supported and the service instance can be consumed by MEC applications running on other MEC hosts, the MEC platform identifies the relevant MEC platforms for this update, and informs them about the changes in service availability by means that may be outside the scope of the present document. The relevant MEC platforms then flag the MEC service instance as running on the other MEC host and send new service notifications to the relevant application instances.
- 5) The MEC application instances, optionally, acknowledge the notification.

In the 2<sup>nd</sup> case MEC service availability update procedure consists of the following steps:

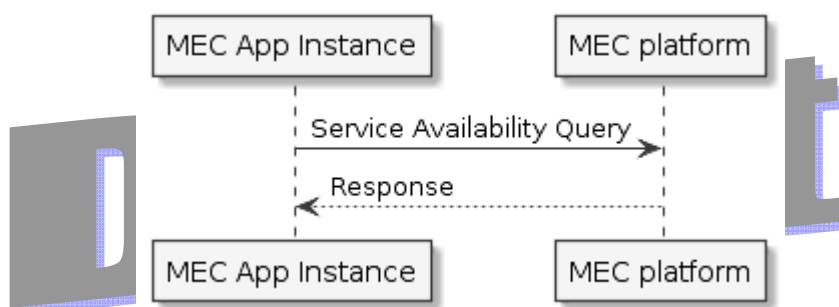
- 1) When a MEC service changes its availability, the service producing MEC application instance sends service availability update message to the MEC platform.
- 2) MEC platform updates the service registry. This step is not to be further specified.
- 3) MEC platform then identifies the relevant MEC application instance for this update (e.g. the applications that indicate the service as "optional" or "required"), and sends service availability notifications to the relevant application instances. If supported and the service can be consumed by MEC applications running on other MEC hosts, the MEC platform identifies the relevant MEC platforms for this update, and informs them about the changes in service availability by means that may be outside the scope of the present document. The relevant MEC platforms then send service availability notifications to the relevant application instances.
- 4) The MEC application instances, optionally, acknowledge the notification.

NOTE 1: In the present document it is not specified on how the MEC platform determines the relevant remote MEC platforms in steps 4 (1<sup>st</sup> case) and step 3 (2<sup>nd</sup> case).

NOTE 2: In the present document it is not specified on how MEC orchestrator is kept informed of the service status updates in remote MEC platforms.

## 5.2.5 Service availability query

Figure 5.2.5-1 shows a scenario where a MEC application instance sends a request to receive information on the availability of a MEC service or a list of MEC services. Typically a MEC application may only query about the MEC service(s) that it has indicated as "optional" or "required".



**Figure 5.2.5-1: Flow of MEC application requesting service availability information**

MEC application requesting service availability information, as illustrated in figure 5.2.5-1, consists of the following steps:

- 1) MEC application instance sends a request to query the availability of a MEC service or a list of MEC services. Typically a MEC application instance may only query about the MEC service(s) that it has indicated as "optional" or "required".
- 2) MEC platform responds with the message body containing the information about the available service(s), including the information needed to access the available service(s). Note that the service availability information is updated by the service producing MEC application instances to the MEC platform.

## 5.2.6 Managing subscription to event notifications

### 5.2.6.1 Introduction

A subscription is required for event notifications that are sent from the MEC platform.

A service availability notification is sent in the following two cases as described in clause 5.2.4:

- When a MEC service is made available by the service producing MEC application, the authorized relevant applications (e.g. the applications that indicate the services as "optional" or "required") will be notified about the newly available service.
- The authorized relevant applications will also be notified about the service availability changes.

An application instance terminate/stop notification is sent in the following two cases as described in clause 5.2.3:

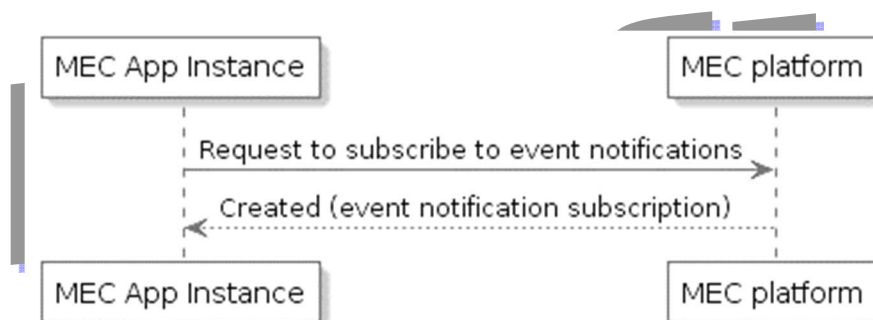
- The MEC platform has received a request for graceful termination of a MEC application instance.
- The MEC platform has received a request for graceful stop of a MEC application instance.

This clause describes the sequence diagram of two related procedures:

- Subscribing to event notifications.
- Unsubscribing from event notifications.

### 5.2.6.2 Subscribing to event notifications

Figure 5.2.6.2-1 shows the message flow for subscribing to event notifications.



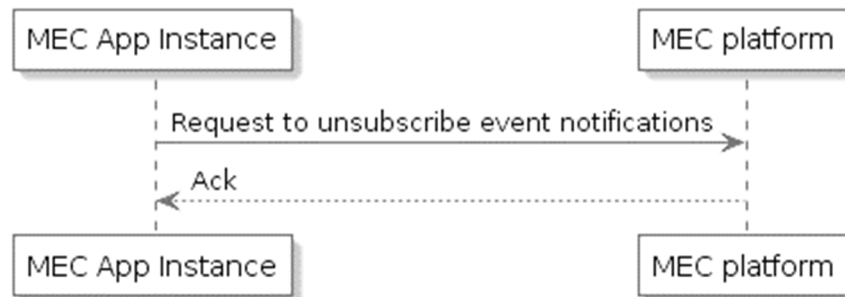
**Figure 5.2.6.2-1: Flow of Subscribing to event notifications**

MEC application requesting event notifications subscription consists of the following steps:

- 1) MEC application instance sends a request to subscribe to event notifications. In case of service availability event notifications, typically a MEC application instance may only subscribe to availability event notifications of the MEC service(s) that it has indicated as "optional" or "required".
- 2) MEC platform responds with the message body containing the created subscription to the event notifications.

### 5.2.6.3 Unsubscribing from event notifications

Figure 5.2.6.3-1 shows the message flow for unsubscribing from event notifications.



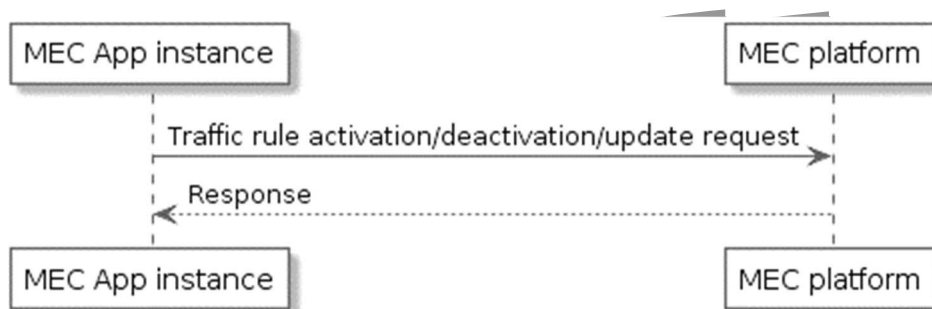
**Figure 5.2.6.3-1: Flow of unsubscribing from event notifications**

MEC application requesting to unsubscribe from the event notifications consists of the following steps:

- 1) MEC application instance sends a request to unsubscribe from the event notifications.
- 2) MEC platform responds with an acknowledgement.

### 5.2.7 Traffic rule activation/deactivation/update

Figure 5.2.7-1 shows a flow for traffic rule activation, deactivation, and update. The MEC application instance may request the MEC platform to activate or deactivate a traffic rule(s). The MEC application instance may request the MEC platform to update the parameters of an existing traffic rule(s).



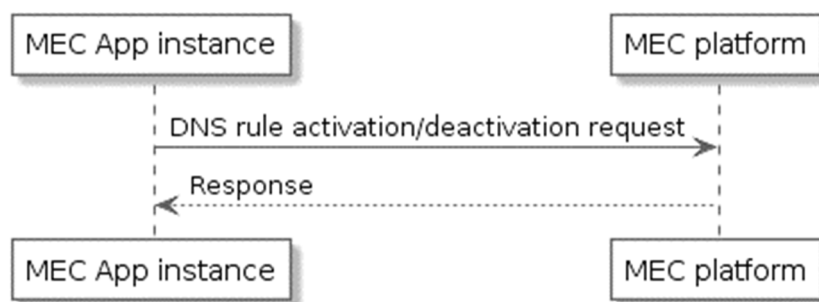
**Figure 5.2.7-1: Flow of traffic rule activation/deactivation/update**

Traffic rule activation/deactivation/update flow consists of the following steps:

- 1) The MEC application instance sends traffic rule activation/deactivation/update request to the MEC platform. The message identifies one or multiple traffic rules that will be activated, deactivated or updated. If the request is authorized, the MEC platform may update the data plane via Mp2 reference point, which is out of the scope of the present document.
- 2) The MEC platform sends response to the MEC application instance to indicate the results of the operation.

### 5.2.8 DNS rule activation/deactivation

Figure 5.2.8-1 shows a DNS rule activation/deactivation flow. The MEC application instance may request the MEC platform to activate or deactivate a DNS rule(s). If the request is authorized and the MEC platform succeeds in finding, based on the information contained in the request, the corresponding DNS rule(s) that have been pre-configured and authenticated by the MEC management, the MEC platform will install or remove the DNS rule(s) into or from the DNS server/proxy.



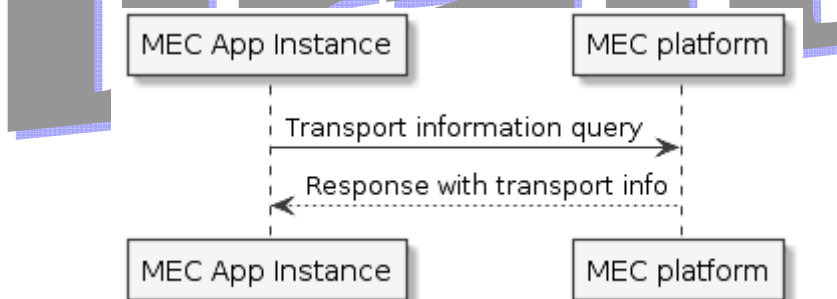
**Figure 5.2.8-1: Flow of DNS rule activation/deactivation**

DNS activation/deactivation flow consists of the following steps:

- 1) The MEC application instance sends DNS activation/deactivation request to the MEC platform. The request includes the DNS rule(s) to be activated or deactivated. If the request is authorized and the MEC platform succeeds in finding, based on the information contained in the request, the corresponding DNS rule(s) that have been pre-configured and authenticated by the MEC management, the platform will install or remove the DNS rule(s) from the DNS server/proxy.
- 2) The MEC platform sends response to the MEC application instance. The response contains the result (success or failure) of the DNS rule activation/deactivation.

## 5.2.9 Transport information query

Providing a MEC service implies the use of a transport to deliver it to the MEC applications that consume it. Examples of transports are REST-HTTP, and message passing systems that support the Publish-Subscribe mode for the communication between MEC application instances and the MEC platform, or between MEC application instances. Any transport other than REST-HTTP is not further specified in the present document. However, transport information query provides a standardized means to the applications to discover the available transports. Figure 5.2.9-1 shows a scenario where the MEC application instance sends a request to receive information on available transports.



**Figure 5.2.9-1: Flow of MEC application requesting transport information**

MEC application instance requesting transport information, as illustrated in figure 5.2.9-1, consists of the following steps:

- 1) MEC application instance sends a request to query the information about transports provided by the platform.
- 2) MEC platform responds with the message body containing the transports information.

## 5.2.10 Time of Day (ToD)

### 5.2.10.1 Introduction

MEC applications may require TOD information for notifications, logs and special events time notions, packets receipt and transmit timestamping and other needs depending on application purpose.

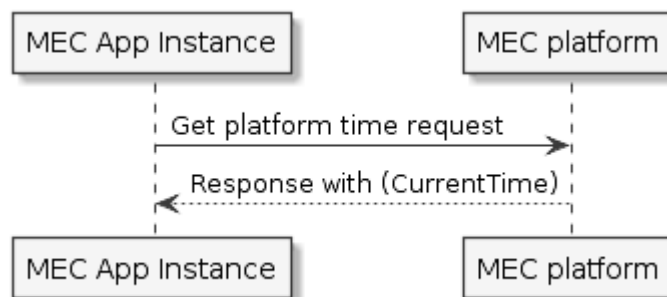
Required TOD accuracy strongly depends on the application itself. Low accuracy TOD information may be provided to the application by use of simple procedure of current time retrieval from the platform. Higher TOD accuracy may be achieved by use of special protocols that allows timing transfer over packet networks, such as NTP specified in IETF RFC 5905 [i.1] or PTP specified in IEEE 1588™ [i.2]. In case of use of packet timing protocols it is assumed that a MEC application will run NTP client or PTP client (referred to as "slave" in [i.2]) while the NTP server(s) or PTP server(s) (referred to as "masters" in [i.2]) may run either by the MEC platform itself or by other facilities for which the application has network connectivity.

This clause specifies two TOD related information exchange flows:

- "Get platform time" flow to get MEC platform current time of day.
- Optional "Timing capabilities query" flow to retrieve information regarding available packet timing facilities.

### 5.2.10.2 Get platform time

Figure 5.2.10.2-1 shows the flow for getting platform time.



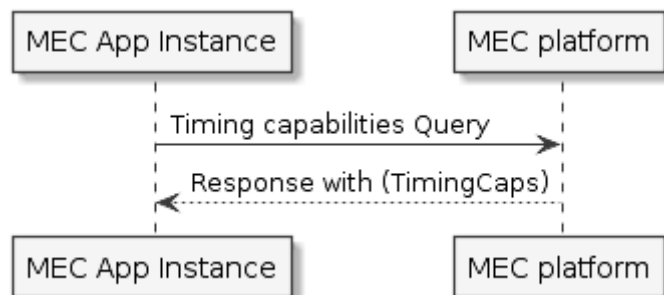
**Figure 5.2.10.2-1: Flow of MEC application requesting platform time**

Get platform time flow consists of the following steps:

- 1) The MEC application instance sends the get platform time request to the MEC platform.
- 2) MEC platform responds with the message body containing CurrentTime.

### 5.2.10.3 Timing capabilities query flow

Figure 5.2.10.3-1 shows a flow for timing capabilities query.



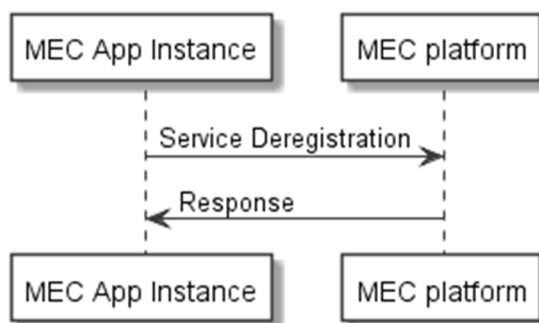
**Figure 5.2.10.3-1: Flow of timing capabilities query**

Timing capabilities query flow consists of the following steps:

- 1) The MEC application instance sends the timing capabilities query request to the MEC platform.
- 2) MEC platform responds with the message body containing TimingCaps.

## 5.2.11 Service deregistration

Figure 5.2.11-1 shows a scenario where a MEC application instance that provides MEC service(s) sends a service deregistration request to the MEC platform.



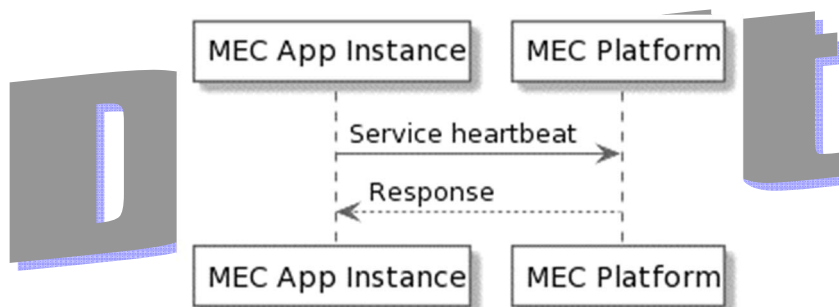
**Figure 5.2.11-1: Flow of MEC service deregistration**

MEC service registration, as illustrated in figure 5.2.11-1, consists of the following steps:

- 1) MEC application instance sends a request to the MEC platform to deregister the MEC service it provides.
- 2) The MEC platform deregisters the MEC service and returns a service deregistration acknowledgement.

## 5.2.12 Service heartbeat

Figure 5.2.12-1 shows a scenario where a MEC service instance sends a heartbeat message to the MEC platform.



**Figure 5.2.12-1: Flow of MEC service deregistration**

Each MEC service instance that has previously registered in MEC platform and is configured for heartbeat sends heartbeat message to the MEC platform periodically in order to show that the MEC service instance is still operational.

The time interval at which the MEC platform is contacted is deployment-specific, and is indicated by the MEC platform to the MEC service instance in a successful service registration.

When the MEC platform has not received the heartbeat for a configurable amount of time, the MEC platform considers that the service instance can no longer be discovered. The MEC platform notifies the MEC service consumers subscribed to receive notifications of status change of the MEC service instance.

MEC service heartbeat, as illustrated in figure 5.2.12-1, consists of the following steps:

- 1) MEC service instance sends a heartbeat message to the MEC platform periodically.
- 2) The MEC platform returns a service heartbeat acknowledgement.

## 5.2.13 MEC application registration

### 5.2.13.1 Introduction

This set of procedures is optional, i.e. it is up to application developer to decide if registration is necessary.

The application registration procedure allows a MEC application instance to provide its information to the MEC platform.

NOTE: The application needs to be instantiated before it can start registration procedure.

Editor's note: further work could be needed to better specify the enabling of discovering via registration. Also further impact on Mx2 is FFS.

If there is a change in the requirements or to the information of an MEC application instance, the MEC application instance uses the application registration update procedure to update the MEC platform.

The MEC application instance uses the application de-registration procedure to remove its information from the MEC platform.

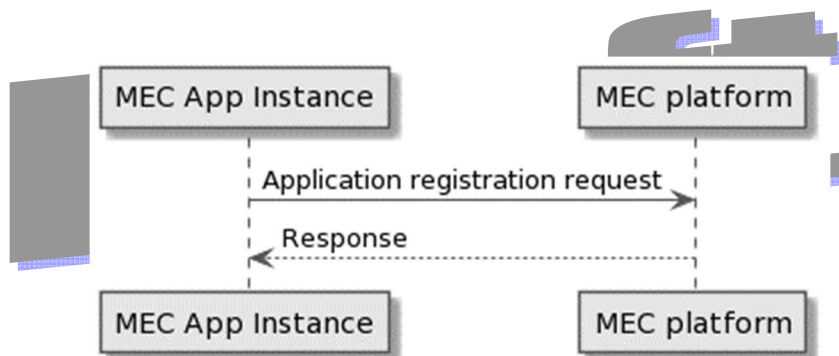
This clause specifies three MEC application registration related information flows:

- "Application registration" flow to register an MEC application instance.

Editor's note: Application registration update and Application de-registration still need to be treated.

### 5.2.13.2 Application registration

Figure 5.2.13.2-1 shows a scenario where a MEC application instance sends an application registration request to the MEC platform.



**Figure 5.2.13.2-1: Flow of MEC application registration**

MEC application registration, as illustrated in Figure 5.2.13.2-1, consists of the following steps:

- 1) MEC application instance sends a request to the MEC platform to register itself.
- 2) The MEC platform registers the MEC application instance and returns an application registration acknowledgement.

Editor's note: It is possible for a MEC application instance to register to both a MEC platform and an Edge Enabler Server (EES). It is still FFS how to specific such a dual application registration, which is an optional registration procedure, i.e. it is up to application developer to decide if dual application registration is necessary.

Editor's note: How to make the MEC application instance aware of the EES to register to is FFS.



## 6 Common data types

### 6.1 Introduction

The following clauses define the data types common to the APIs specified in the present document.

### 6.2 Resource data types

#### 6.2.1 Introduction

This clause defines data structures to be used in resource representations.

#### 6.2.2 Type: SubscriptionLinkList

This type represents a list of links related to currently existing subscriptions for a MEC application instance. This information is returned when sending a request to receive current subscriptions.

**Table 6.2.2-1: Attributes of the SubscriptionLinkList**

Attribute name	Data type	Cardinality	Description
_links	Structure (inlined)	1	Object containing hyperlinks related to the resource.
>self	LinkType	1	Self-referring URI.
>subscriptions	array(Structure (inlined))	0..N	The MEC application instance's subscriptions.
>>href	Uri	1	URI referring to the subscription.
>>subscriptionType	String	1	Type of the subscription. The values are as defined in the "subscriptionType" attribute for each different Mp1 event subscription data type.

### 6.3 Referenced structured data types

#### 6.3.1 Introduction

This clause defines data structures that are referenced from multiple APIs specified in the present document.

#### 6.3.2 Type: LinkType

This type represents a type of link and may be referenced from data structures.

**Table 6.3.2-1: Attributes of the LinkType**

Attribute name	Data type	Cardinality	Description
href	Uri	1	URI referring to a resource

## 7 MEC application support API

### 7.1 Data model

#### 7.1.1 Introduction

Clauses 7.1.2 to 7.1.6 specify the data types that are used to implement the MEC application support API for which the relevant sequence diagrams are described in clause 5.2.

#### 7.1.2 Resource data types

##### 7.1.2.1 Introduction

This clause defines data structures to be used in resource representations.

##### 7.1.2.2 Type: TrafficRule

This type represents the general information of a traffic rule.

The attributes of the TrafficRule shall follow the indications provided in table 7.1.2.2-1.

**Table 7.1.2.2-1: Attributes of TrafficRule**

Attribute name	Data type	Cardinality	Description
trafficRuleId	String	1	Identify the traffic rule.
filterType	Enum (inlined)	1	Definition of filter per FLOW or PACKET. If FLOW, the filter matches upstream (e.g. UE→EPC) packets and upstream (e.g. EPC→UE) packets are handled in the same context.
priority	Int	1	Priority of this traffic rule within the range 0 to 255. If traffic rules conflict, the one with higher priority take precedence. See note 1.
trafficFilter	TrafficFilter	1..N	The filter used to identify specific packets that need to be handled by the MEC host.
action	Enum (inlined)	1	The action of the MEC host data plane when a packet matches the trafficFilter, the following actions are defined: <ul style="list-style-type: none"> <li>• DROP</li> <li>• FORWARD_DECAPSULATED</li> <li>• FORWARD_ENCAPSULATED</li> <li>• PASSTHROUGH</li> <li>• DUPLICATE_DECAPSULATED</li> <li>• DUPLICATE_ENCAPSULATED</li> </ul>
dstInterface	DestinationInterface	0..2	Describes the destination interface information. If the action is FORWARD_DECAPSULATED, FORWARD_ENCAPSULATED or PASSTHROUGH one value shall be provided. If the action is DUPLICATE_DECAPSULATED or DUPLICATE_ENCAPSULATED, two values shall be provided. See note 2. If the action is DROP, no value shall be provided.
state	Enum (inlined)	1	Contains the traffic rule state: ACTIVE, INACTIVE. This attribute may be updated using HTTP PUT method.
NOTE 1: Value indicates the priority in descending order, i.e. with 0 as the highest priority and 255 as the lowest priority.			
NOTE 2: Some applications (like inline/tap) require two interfaces. The first interface in the case of inline/tap is on the client (e.g. UE) side and the second on the core network (e.g. EPC) side.			

### 7.1.2.3 Type: DnsRule

This type represents the general information of a DNS rule.

The attributes of the DnsRule shall follow the indications provided in table 7.1.2.3-1.

**Table 7.1.2.3-1: Attributes of DnsRule**

Attribute name	Data type	Cardinality	Description
dnsRuleId	String	1	Identifies the DNS Rule.
domainName	String	1	FQDN resolved by the DNS rule.
ipAddressType	Enum (inlined)	1	Specify the IP address type, value: IP_V6, IP_V4.
ipAddress	String	1	IP address associated with the FQDN resolved by the DNS rule.
ttl	Int	0..1	Time to live value, in seconds.
state	Enum (inlined)	1	Contains the DNS rule state: ACTIVE, INACTIVE. This attribute may be updated using HTTP PUT method.
NOTE: If no ttl value is provided, the DnsRule shall not expire.			

### 7.1.2.4 Type: TimingCaps

This type represents the information provided by the MEC platform in response to the "Timing capabilities Query" message.

The attributes of the TimingCaps shall follow the indications provided in table 7.1.2.4-1.

**Draft**

Table 7.1.2.4-1: Attributes of TimingCaps

Attribute name	Data type	Cardinality	Description
timeStamp	Structure (inlined)	0..1	
>seconds	Uint32	1	The seconds part of the Time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC
>nanoSeconds	Uint32	1	The nanoseconds part of the Time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC
ntpServers	Structure (inlined)	0..N	Number of available NTP servers
>ntpServerAddrType	Enum (inlined)	1	Address type of NTP server with the following permitted values: <ul style="list-style-type: none"> <li>• IP_ADDRESS</li> <li>• DNS_NAME</li> </ul>
>ntpServerAddr	String	1	NTP server address
>minPollingInterval	Uint32	1	Minimum poll interval for NTP messages, in seconds as a power of two Range: 3 to 17
>maxPollingInterval	Uint32	1	Maximum poll interval for NTP messages, in seconds as a power of two Range: 3 to 17
>localPriority	Uint32	1	NTP server local priority
>authenticationOption	Enum (inlined)	1	NTP authentication option with the following permitted values: <ul style="list-style-type: none"> <li>• NONE</li> <li>• SYMMETRIC_KEY</li> <li>• AUTO_KEY</li> </ul>
>authenticationKeyNum	Uint32	0..1	Authentication key number. This configuration is valid and shall be present if authenticationOption is set to SYMMETRIC_KEY
ptpMasters	Structure (inlined)	0..N	Number of available PTP Servers (referred to as "masters" in IEEE 1588-2019™ [i.2])
>ptpMasterIpAddress	String	1	PTP Server (referred to as "master" in IEEE 1588-2019™ [i.2]) IP Address
>ptpMasterLocalPriority	Uint32	1	PTP Server (referred to as "master" in IEEE 1588-2019™ [i.2]) local priority
>delayReqMaxRate	Uint32	1	Acceptable maximum rate of the Delay_Req messages in packets per second

### 7.1.2.5 Type: CurrentTime

This type represents the information provided by the MEC platform in response to the "Get Platform Time Request" message.

The attributes of the CurrentTime shall follow the indications provided in table 7.1.2.5-1.

Table 7.1.2.5-1: Attributes of CurrentTime

Attribute name	Data type	Cardinality	Description
seconds	Uint32	1	The seconds part of the Time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC
nanoSeconds	Uint32	1	The nanoseconds part of the Time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC
timeSourceStatus	Enum (inlined)	1	Platform Time Source status with the following permitted values: <ul style="list-style-type: none"> <li>• TRACEABLE - time source is locked to the UTC time source</li> <li>• NONTRACEABLE - time source is not locked to the UTC time source</li> </ul>

### 7.1.2.6 Type: AppInfo

This type represents the information provided by the MEC application instance as part of the "application registration request" message.

The attributes of the AppInfo shall follow the indications provided in table 7.1.2.6-1.

**Table 7.1.2.6-1: Attributes of AppInfo**

Attribute name	Data type	Cardinality	Description
appName	String	1	Name of the application. It shall be consistent with the appName in the AppD, if an AppD is available.
appProvider	String	1	Provider of the application. It shall be consistent with the appProvider in the AppD, if an AppD is available. See note 1.
appCategory	CategoryRef	0..1	Category of the application.
appDId	String	0..1	The application descriptor identifier. It is managed by the application provider to identify the application descriptor in a globally unique way. Shall be present if the application instance is instantiated by the MEC Management.
appInstanceId	String	0..1	Identifier of the application instance. Shall be present if the application instance is instantiated by the MEC Management.  <b>Editor's note #1: further clarification is needed to appInstanceId.</b>
endpoint	EndPointInfo	0..1	Endpoint information (e.g. URI, FQDN, IP address) of the application server, which is part of the application functionalities. See note 2.
appServiceRequired	ServiceDependency	0..N	Describes services a MEC application requires to run. ServiceDependency is defined in ETSI GS MEC 010-2 [4]. It shall shall not be provided if an AppD is available.
appServiceOptional	ServiceDependency	0..N	Describes services a MEC application may use if available. ServiceDependency is defined in ETSI GS MEC 010-2 [4]. It shall shall not be provided if an AppD is available.
appFeatureRequired	FeatureDependency	0..N	Describes features a MEC application requires to run. FeatureDependency is defined in ETSI GS MEC 010-2 [4]. It shall shall not be provided if an AppD is available.
appFeatureOptional	FeatureDependency	0..N	Describes features a MEC application may use if available. FeatureDependency is defined in ETSI GS MEC 010-2 [4]. It shall shall not be provided if an AppD is available.
isEas	Boolean	0..1	Indicate whether the application is an EAS (as defined in ETSI TS 123 558 [21]).  Default to FALSE if absent.
easProfile	EASProfile	0..1	EAS profile as defined in 3GPP TS 29.558 [19]. Shall be present if isEas is TRUE. See note 1 and note 2.
NOTE 1: If EASProfile is present, provId shall be consistent with appProvider, i.e. the same.			
NOTE 2: If EASProfile is present, endpoint shall refer to the same end point as endPt provided in that data type.			

**Editor's note #2: it is FFS if other attributes in the AppD should be added.**

**Editor's note #3: further attributes may be added to the AppInfo data structure.**

## 7.1.3 Subscription data types

### 7.1.3.1 Introduction

This clause defines data structures that define criteria to be used in subscriptions.

### 7.1.3.2 Type: AppTerminationNotificationSubscription

This type represents a subscription to the notifications from the MEC platform related to MEC application instance termination/stop.

The attributes of the AppTerminationNotificationSubscription shall follow the indications provided in table 7.1.3.2-1.

**Table 7.1.3.2-1: Attributes of AppTerminationNotificationSubscription**

Attribute name	Data type	Cardinality	Description
subscriptionType	String	1	Shall be set to "AppTerminationNotificationSubscription".
callbackReference	Uri	1	URI selected by the MEC application instance to receive notifications on the subscribed MEC application instance management information. This shall be included in both the request and the response.
_links	Structure (inlined)	0..1	Object containing hyperlinks related to the resource. This shall only be included in the HTTP responses.
>self	LinkType	1	Self-referring URI.
appInstanceIcd	String	1	It is used as the filtering criterion for the subscribed events.

## 7.1.4 Notification data types

### 7.1.4.1 Introduction

This clause defines data structures that define notifications.

### 7.1.4.2 Type: AppTerminationNotification

This type represents the information that the MEC platform notifies the subscribed application instance about the corresponding application instance termination/stop.

The attributes of the AppTerminationNotification shall follow the indications provided in table 7.1.4.2-1.

**Table 7.1.4.2-1: Attributes of AppTerminationNotification**

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Shall be set to "AppTerminationNotification".
operationAction	Enum (inlined)	1	Operation that is being performed on the MEC application instance: <ul style="list-style-type: none"> <li>• STOPPING</li> <li>• TERMINATING</li> </ul>
maxGracefulTimeout	Uint32	1	Maximum non-zero timeout value in seconds for graceful termination or graceful stop of an application instance.
_links	Structure (inlined)	1	Object containing hyperlinks related to the resource.
>subscription	LinkType	1	A link to the related subscription.
>confirmTermination	LinkType	0..1	Link to the task resource where to confirm termination/stop in case the application is ready to be terminated, or to be considered stopped by the MEC Platform, before expiry of the timeout.

### 7.1.4.3 Type: AppTerminationConfirmation

This type represents the information that the MEC application instance provides to the MEC platform when informing it that the application has completed its application level related terminate/stop actions, e.g. retention of application state in the case of stop.

The attributes of the AppTerminationConfirmation type shall follow the indications provided in table 7.1.4.3-1.

**Table 7.1.4.3-1: Attributes of AppTerminationConfirmation**

Attribute name	Data type	Cardinality	Description
operationAction	Enum (inlined)	1	Operation that is being performed on the MEC application instance: <ul style="list-style-type: none"> <li>• STOPPING</li> <li>• TERMINATING</li> </ul> The value shall match that sent in the corresponding AppTerminationNotification.

### 7.1.4.4 Type: AppReadyConfirmation

This type represents the information that the MEC application instance indicates to the MEC platform that it is up and running.

The attributes of the AppReadyConfirmation type shall follow the indications provided in table 7.1.4.4-1.

**Table 7.1.4.4-1: Attributes of AppReadyConfirmation**

Attribute name	Data type	Cardinality	Description
indication	String	1	Indication about the MEC application instance: <ul style="list-style-type: none"> <li>• READY</li> </ul>

## 7.1.5 Referenced structured data types

### 7.1.5.1 Introduction

This clause defines data structures that may be referenced from data structures defined in clauses 7.1.2 to 7.1.4, but are neither resource representations nor notifications.

### 7.1.5.2 Type: TrafficFilter

This type represents the traffic filter.

The attributes of the TrafficFilter shall follow the indications provided in table 7.1.5.2-1.

Table 7.1.5.2-1: Attributes of TrafficFilter

Attribute name	Data type	Cardinality	Description
srcAddress	String	0..N	An IP address or a range of IP address. For IPv4, the IP address could be an IP address plus mask, or an individual IP address, or a range of IP addresses. For IPv6, the IP address could be an IP prefix, or a range of IP prefixes.
dstAddress	String	0..N	An IP address or a range of IP address. For IPv4, the IP address could be an IP address plus mask, or an individual IP address, or a range of IP addresses. For IPv6, the IP address could be an IP prefix, or a range of IP prefixes.
srcPort	String	0..N	A port or a range of ports.
dstPort	String	0..N	A port or a range of ports.
protocol	String	0..N	Specify the protocol of the traffic filter.
tag	String	0..N	Used for tag based traffic rule.
srcTunnelAddress	String	0..N	Used for GTP tunnel based traffic rule.
tgtTunnelAddress	String	0..N	Used for GTP tunnel based traffic rule.
srcTunnelPort	String	0..N	Used for GTP tunnel based traffic rule.
dstTunnelPort	String	0..N	Used for GTP tunnel based traffic rule.
qCl	Int	0..1	Used to match all packets that have the same QCI.
dSCP	Int	0..1	Used to match all IPv4 packets that have the same DSCP.
tC	Int	0..1	Used to match all IPv6 packets that have the same TC.

### 7.1.5.3 Type: DestinationInterface

This type represents the destination interface.

The attributes of the DestinationInterface shall follow the indications provided in table 7.1.5.3-1.

Table 7.1.5.3-1: Attributes of DestinationInterface

Attribute name	Data type	Cardinality	Description
interfaceType	Enum (inlined)	1	Type of the interface, e.g. TUNNEL, MAC, IP, etc.
tunnelInfo	TunnelInfo	0..1	Included only if the destination interface type is "tunnel".
srcMacAddress	String	0..1	If the interface type is "MAC", source address identifies the MAC address of the interface.
dstMacAddress	String	0..1	If the interface type is "MAC", destination address identifies the MAC address of the interface. Only used for dstInterface.
dstIpAddress	String	0..1	If the interface type is "IP", destination address identifies the IP address of the remote destination. Only used for dstInterface.

### 7.1.5.4 Type: TunnelInfo

This type represents the tunnel information.

The attributes of the TunnelInfo shall follow the indications provided in table 7.1.5.4-1.

Table 7.1.5.4-1: Attributes of TunnelInfo

Attribute name	Data type	Cardinality	Description
tunnelType	Enum (inlined)	1	Type of the tunnel, e.g. GTP_U, GRE, etc.
tunnelDstAddress	String	0..1	Destination address of the tunnel
tunnelSrcAddress	String	0..1	Source address of the tunnel



## 7.1.6 Referenced simple data types and enumerations

Neither simple data types nor enumerations are defined for this API.

## 7.2 API definition

### 7.2.1 Introduction

This clause defines the resources and operations of the MEC application support API.

### 7.2.2 Global definitions and resource structure

All resource URIs of this API shall have the following root:

- **{apiRoot}/{apiName}/{apiVersion}/**

The "apiRoot" is discovered using the service registry. The "apiName" shall be set to "mec\_app\_support" and the "apiVersion" shall be set to "v1" for the present document. It includes the scheme ("https"), host and optional port, and an optional prefix string. All resource URIs in clauses 7.2.3 to 7.2.11 are defined relative to the above root URI.

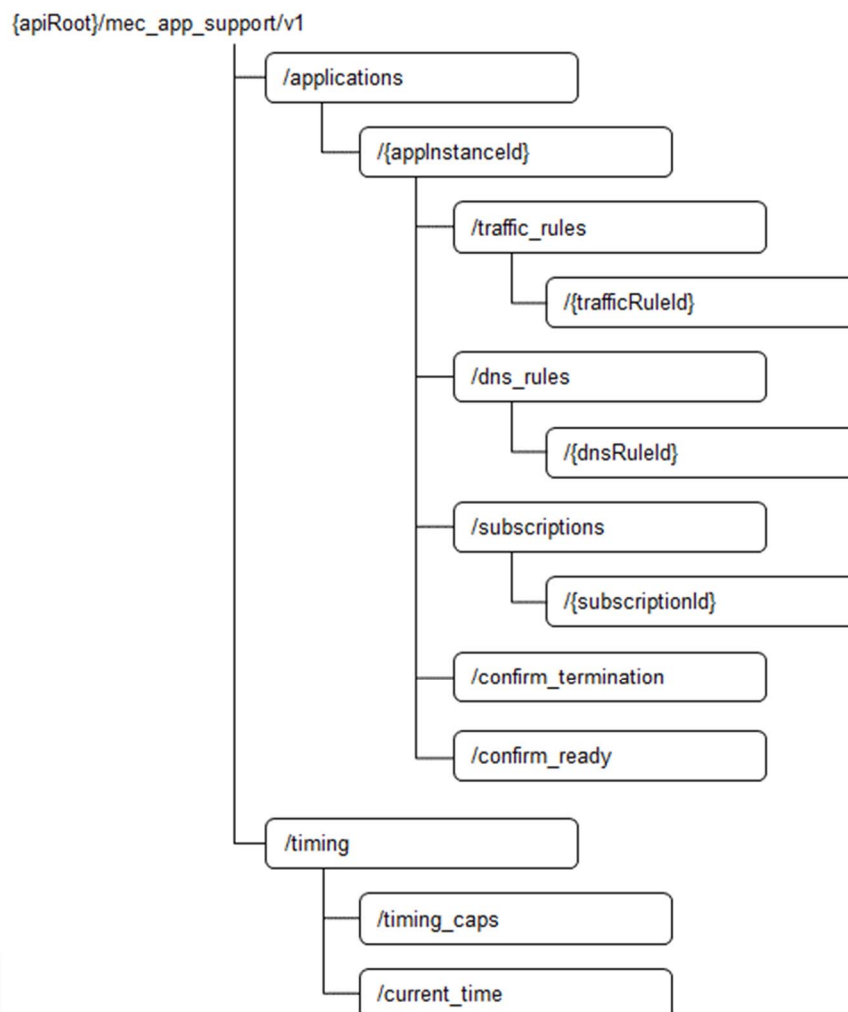
The API shall support HTTP over TLS (also known as HTTPS) using TLS version 1.2 (as defined by IETF RFC 5246 [7]). TLS 1.3 (including the new specific requirements for TLS 1.2 implementations) defined by IETF RFC 8446 [18] should be supported. HTTP without TLS shall not be used. Versions of TLS earlier than 1.2 shall neither be supported nor used.

This API shall require the use of the OAuth 2.0 client credentials grant type according to IETF RFC 6749 [13] with bearer tokens according to IETF RFC 6750 [14]. See clause 6.16 of ETSI GS MEC 009 [5] for more information. How the token endpoint and client credentials are provisioned into the MEC applications is out of scope of the present document.

This API supports additional application-related error information to be provided in the HTTP response when an error occurs. See clause 6.15 of ETSI GS MEC 009 [5] for more information.

Figure 7.2.2-1 illustrates the resource URI structure of this API.

**Draft**



**Figure 7.2.2-1: Resource URI structure of the MEC application support API**

Table 7.2.2-1 provides an overview of the resources defined by the present specification for the MEC application support API, and the applicable HTTP methods.

**Table 7.2.2-1: Resources and methods overview**

Resource name	Resource URI	HTTP method	Meaning
Parent resource of all mecAppSupportSubscription of a subscriber	/applications/{applInstancelid}/subscriptions	GET	Retrieve information about a list of mecAppSupportSubscription resources for this subscriber
		POST	Create a mecAppSupportSubscription resource
Individual mecAppSupportSubscription	/applications/{applInstancelid}/subscriptions/{subscriptionId}	GET	Retrieve information about a mecAppSupportSubscription resource for this subscriber
		DELETE	Delete a mecAppSupportSubscription resource
Parent resource of all mecTrafficRule of an application instance	/applications/{applInstancelid}/traffic_rules	GET	Retrieve information about a list of mecTrafficRule resources for an application instance

Resource name	Resource URI	HTTP method	Meaning
Individual mecTrafficRule	/applications/{appInstanceId}/traffic_rules/{trafficRuleId}	GET	Retrieve information about a mecTrafficRule resource
		PUT	Update the information about a mecTrafficRule resource
Parent resource of all mecDnsRule of an application instance	/applications/{appInstanceId}/dns_rules	GET	Retrieve information about a list of mecDnsRule resources for an application instance
Individual mecDnsRule	/applications/{appInstanceId}/dns_rules/{dnsRuleId}	GET	Retrieve information about a mecDnsRule resource
		PUT	Update the information about a mecDnsRule resource
confirm termination task	/applications/{appInstanceId}/confirm_termination	POST	Confirm the application level termination of an App instance
confirm ready task	/applications/{appInstanceId}/confirm_ready	POST	Confirm the application instance is up and running
mecTimingCaps	/timing/timing_caps	GET	Retrieve information about the mecTimingCaps resource
mecCurrentTime	/timing/current_time	GET	Retrieve information about the mecCurrentTime resource

## 7.2.3 Resource: all mecAppSupportSubscription

### 7.2.3.1 Description

This resource is used to represent all subscriptions of a subscriber to the notifications from the MEC platform.

### 7.2.3.2 Resource definition

Resource URI: {apiRoot}/mec\_app\_support/v1/applications/{appInstanceId}/subscriptions

Resource URI variables for this resource are defined in table 7.2.3.2-1.

**Table 7.2.3.2-1: Resource URI variables for resource "all mecAppSupportSubscription"**

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.

### 7.2.3.3 Resource methods

#### 7.2.3.3.1 GET

The GET method may be used to request information about all subscriptions for this requestor. Upon success, the response contains payload body with all the subscriptions for the requestor.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.3.3.1-1 and 7.2.3.3.1-2.

**Table 7.2.3.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

Table 7.2.3.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	SubscriptionLinkList	1	200 OK	Upon success, a response body containing the list of links to the requested subscriptions is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 7.2.3.3.2 PUT

Not supported.

## 7.2.3.3.3 PATCH

Not supported.

## 7.2.3.3.4 POST

The POST method may be used to create a new subscription. One example use case is to create a new subscription to the MEC application termination notifications. Upon success, the response contains payload body describing the created subscription.

POST HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in the tables 7.2.3.3.4-1 and 7.2.3.3.4-2.

Table 7.2.3.3.4-1: URI query parameters supported by the POST method on this resource

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.3.3.4-2: Data structures supported by the POST request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	AppTerminationNotificationSubscription	1	Payload body in the request contains a subscription to the MEC application termination notifications that is to be created.	
Response body	Data type	Cardinality	Response codes	Remarks
	AppTerminationNotificationSubscription	1	201 Created	Upon success, the HTTP response shall include a "Location" HTTP header that contains the resource URI of the created subscription resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.	

### 7.2.3.3.5 DELETE

Not supported.

## 7.2.4 Resource: individual mecAppSupportSubscription

### 7.2.4.1 Description

This resource is used to represent a subscription to the notifications from the MEC platform. When this resource represents a subscription to the notifications related to MEC application instance termination/stop, it shall follow the data type of "AppTerminationNotificationSubscription" as specified in clause 7.1.3.2. The notifications that are related to a AppTerminationNotificationSubscription shall follow the data type of "AppTerminationNotification" as specified in clause 7.1.4.2.

### 7.2.4.2 Resource definition

Resource URI: {apiRoot}/mec\_app\_support/v1/applications/{appInstanceId}/subscriptions/{subscriptionId}

Resource URI variables for this resource are defined in table 7.2.4.2-1.

**Table 7.2.4.2-1: Resource URI variables for resource "individual mecAppSupportSubscription"**

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.
subscriptionId	Represents a subscription to the notifications from the MEC platform.

### 7.2.4.3 Resource methods

#### 7.2.4.3.1 GET

The GET method requests information about a subscription for this requestor. Upon success, the response contains payload body with the subscription for the requestor.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.4.3.1-1 and 7.2.4.3.1-2.

**Table 7.2.4.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.4.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	AppTerminationNotification Subscription	1	200 OK	Upon success, a response body containing the requested subscription is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 7.2.4.3.2

PUT

Not supported.

#### 7.2.4.3.3

PATCH

Not supported.

#### 7.2.4.3.4

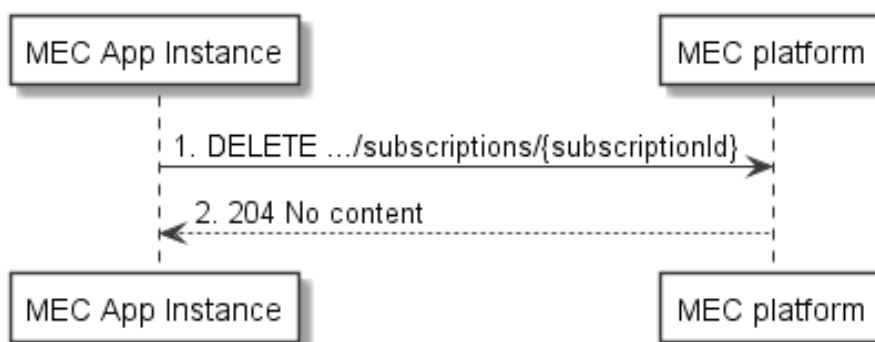
POST

Not supported.

#### 7.2.4.3.5

DELETE

This method deletes a mecAppSupportSubscription. This method is typically used in "Unsubscribing from event notifications" procedure as described in clause 5.2.6.3. Figure 7.2.4.3.5-1 shows the example message flows using DELETE method.



**Figure 7.2.4.3.5-1: Unsubscribing from MEC application support event notifications**

DELETE HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.4.3.5-1 and 7.2.4.3.5-2.

**Table 7.2.4.3.5-1: URI query parameters supported by the DELETE method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.4.3.5-2: Data structures supported by the DELETE request on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	n/a		204 No Content	
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 7.2.5 Resource: mecTimingCaps

### 7.2.5.1 Description

This resource is used to represent the timing capabilities of the MEC platform.

### 7.2.5.2 Resource definition

Resource URI: **{apiRoot}/mec\_app\_support/v1/timing/timing\_caps**

Resource URI variables for this resource are defined in table 7.2.5.2-1.

**Table 7.2.5.2-1: Resource URI variables for resource "mecTimingCaps"**

Name	Definition
apiRoot	See clause 7.2.2

### 7.2.5.3 Resource methods

#### 7.2.5.3.1 GET

This method retrieves the information of the platform's timing capabilities which corresponds to the timing capabilities query as described in clause 5.2.10.3. Figure 7.2.5.3.1-1 shows the example message flow for retrieving timing capabilities using GET method.



**Figure 7.2.5.3.1-1: GET timing capabilities flow**

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.5.3.1-1 and 7.2.5.3.1-2.

**Table 7.2.5.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.5.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	TimingCaps	1	200 OK	It is used to indicate nonspecific success. The response body contains a representation of the resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden		The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 7.2.5.3.2 PUT

Not supported.

#### 7.2.5.3.3 PATCH

Not supported.



## 7.2.5.3.4 POST

Not supported.

## 7.2.5.3.5 DELETE

Not supported.

## 7.2.6 Resource: mecCurrentTime

## 7.2.6.1 Description

This resource is used to represent the current time of the MEC platform.

## 7.2.6.2 Resource definition

Resource URI: `{apiRoot}/mec_app_support/v1/timing/current_time`

Resource URI variables for this resource are defined in table 7.2.6.2-1.

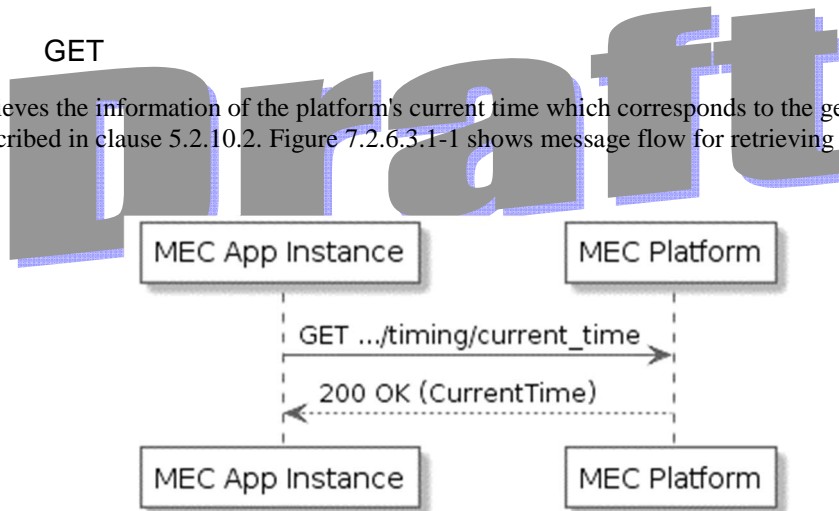
**Table 7.2.6.2-1: Resource URI variables for resource "mecCurrentTime"**

Name	Definition
apiRoot	See clause 7.2.2.

## 7.2.6.3 Resource methods

## 7.2.6.3.1 GET

This method retrieves the information of the platform's current time which corresponds to the get platform time procedure as described in clause 5.2.10.2. Figure 7.2.6.3.1-1 shows message flow for retrieving current time using GET method.



**Figure 7.2.6.3.1-1: GET platform time API flow**

This method shall comply with the URI query parameters, request and response data structures, as specified in tables 7.2.6.3.1-1 and 7.2.6.3.1-2.

**Table 7.2.6.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

Table 7.2.6.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response Codes	Remarks
	CurrentTime	1	200 OK	It is used to indicate nonspecific success. The response body contains a representation of the resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 7.2.6.3.2 PUT

Not supported.

## 7.2.6.3.3 PATCH

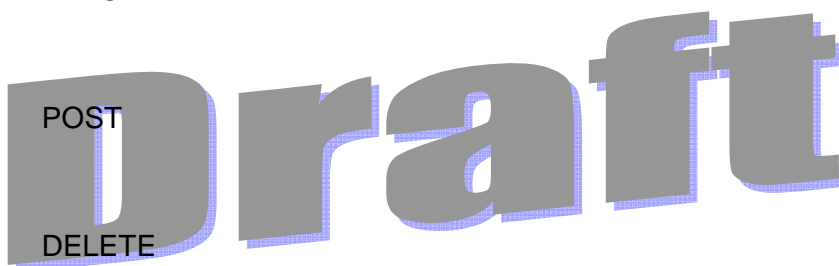
Not supported.

## 7.2.6.3.4 POST

Not supported.

## 7.2.6.3.5 DELETE

Not supported.



## 7.2.7 Resource: all mecTrafficRule

## 7.2.7.1 Description

This resource is used to represent all the traffic rules associated with a MEC application instance, which follows the resource data type of "TrafficRule" as specified in clause 7.1.2.2.

## 7.2.7.2 Resource definition

Resource URI: {apiRoot}/mec\_app\_support/v1/applications/{appInstanceId}/traffic\_rules

Resource URI variables for this resource are defined in table 7.2.7.2-1.

Table 7.2.7.2-1: Resource URI variables for resource "all mecTrafficRule"

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.

### 7.2.7.3 Resource methods

#### 7.2.7.3.1 GET

This method retrieves information about all the traffic rules associated with a MEC application instance.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.7.3.1-1 and 7.2.7.3.1-2.

**Table 7.2.7.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.7.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	TrafficRule	0..N	200	Upon success, a response body containing an array of the TrafficRules is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden		The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 7.2.7.3.2 PUT

Not supported.

#### 7.2.7.3.3 PATCH

Not supported.

#### 7.2.7.3.4 POST

Not supported.

#### 7.2.7.3.5 DELETE

Not supported.

## 7.2.8 Resource: individual mecTrafficRule

### 7.2.8.1 Description

This resource is used to represent a traffic rule, which follows the resource data type of "TrafficRule" as specified in clause 7.1.2.2.

## 7.2.8.2 Resource definition

Resource URI: `{apiRoot}/mec_app_support/v1/applications/{appInstanceId}/traffic_rules/{trafficRuleId}`

Resource URI variables for this resource are defined in table 7.2.8.2-1.

**Table 7.2.8.2-1: Resource URI variables for resource "individual mecTrafficRule"**

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.
trafficRuleId	Represents a traffic rule.

## 7.2.8.3 Resource methods

### 7.2.8.3.1 GET

This method retrieves information about a traffic rule associated with a MEC application instance.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.8.3.1-1 and 7.2.8.3.1-2.

**Table 7.2.8.3.1-1: URI query parameters supported by the GET method on this resource**

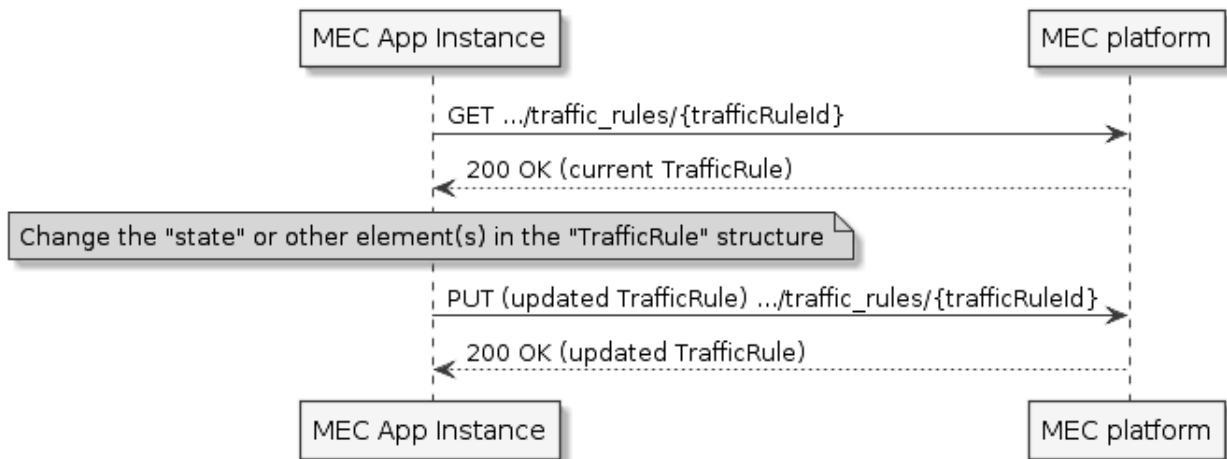
Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.8.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	TrafficRule	1	200	Upon success, a response body containing the TrafficRules is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.	

### 7.2.8.3.2 PUT

This method activates, de-activates or updates a traffic rule. Figure 7.2.8.3.2-1 shows the message flow of "Traffic rule activation/deactivation/update" using PUT.



**Figure 7.2.8.3.2-1: Traffic rule activation/deactivation/update using PUT**

PUT HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.8.3.2-1 and 7.2.8.3.2-2.

**Table 7.2.8.3.2-1: URI query parameters supported by the PUT method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.8.3.2-2: Data structures supported by the PUT request/response on this resource**

	Data type	Cardinality	Remarks	
	Request body	TrafficRule	1	One or more updated attributes that are allowed to be changed (i.e. "state" or other attributes based on definition in clause 7.1.2.2) are included in the TrafficRule data structure in the payload body of the request.
Response body	Data type	Cardinality	Response codes	Remarks
	TrafficRule	1	200 OK	Upon success, a response body containing data type describing the updated TrafficRule is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
ProblemDetails	0..1	412 Precondition Failed	It is used when a condition has failed during conditional requests, e.g. when using ETags to avoid write conflicts. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.	

### 7.2.8.3.3 PATCH

Not supported.

## 7.2.8.3.4 POST

Not supported.

## 7.2.8.3.5 DELETE

Not supported.

## 7.2.9 Resource: all mecDnsRule

## 7.2.9.1 Description

This resource is used to represent all the DNS rules associated with a MEC application instance, which follows the resource data type of "DnsRule" as specified in clause 7.1.2.3.

## 7.2.9.2 Resource definition

Resource URI: **{apiRoot}/mec\_app\_support/v1/applications/{appInstanceId}/dns\_rules**

Resource URI variables for this resource are defined in table 7.2.9.2-1.

**Table 7.2.9.2-1: Resource URI variables for resource "all mecDnsRule"**

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.

## 7.2.9.3 Resource methods

## 7.2.9.3.1 GET

This method retrieves information about all the DNS rules associated with a MEC application instance.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.9.3.1-1 and 7.2.9.3.1-2.

**Table 7.2.9.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

Table 7.2.9.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	DnsRule	0..N	200 OK	Upon success, a response body containing an array of the DnsRules is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 7.2.9.3.2 PUT

Not supported.

## 7.2.9.3.3 PATCH

Not supported.

## 7.2.9.3.4 POST

Not supported.

## 7.2.9.3.5 DELETE

Not supported.

**Draft**

## 7.2.10 Resource: individual mecDnsRule

## 7.2.10.1 Description

This resource is used to represent a DNS rule, which follows the resource data type of "DnsRule" as specified in clause 7.1.2.3.

## 7.2.10.2 Resource definition

Resource URI: `{apiRoot}/mec_app_support/v1/applications/{appInstanceId}/dns_rules/{dnsRuleId}`

Resource URI variables for this resource are defined in table 7.2.10.2-1.

Table 7.2.10.2-1: Resource URI variables for resource "individual mecDnsRule"

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance. Note the appInstanceId is allocated by the MEC platform manager.
dnsRuleId	Represents a DNS rule.

### 7.2.10.3 Resource methods

#### 7.2.10.3.1 GET

This method retrieves information about a DNS rule associated with a MEC application instance.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.10.3.1-1 and 7.2.10.3.1-2.

**Table 7.2.10.3.1-1: URI query parameters supported by the GET method on this resource**

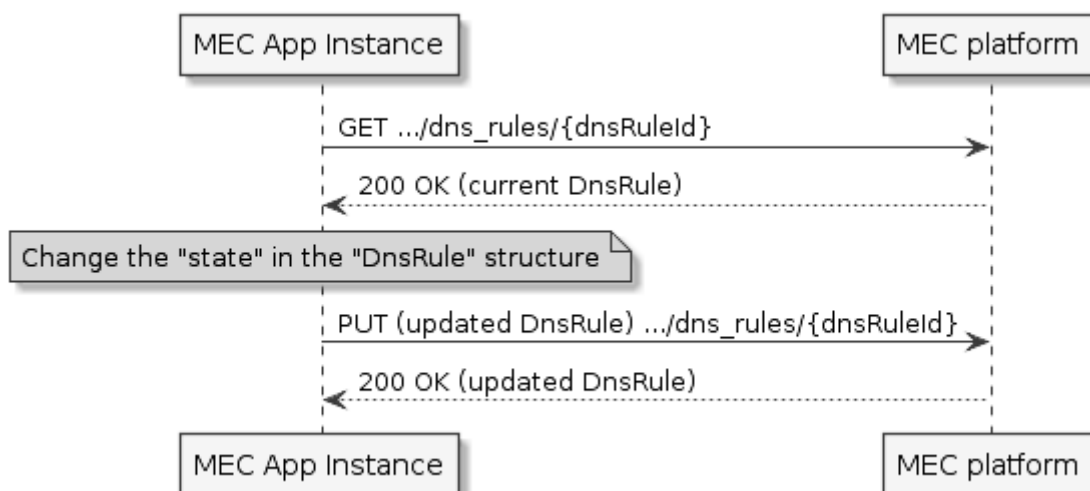
Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.10.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	DnsRule	1	200 OK	Upon success, a response body containing the DnsRules is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden		The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 7.2.10.3.2 PUT

This method activates, de-activates or updates a DNS rule. Figure 7.2.10.3.2-1 shows the message flow of "DNS rule activation/deactivation" using PUT.



**Figure 7.2.10.3.2-1: DNS rule activation/deactivation using PUT**



PUT HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.10.3.2-1 and 7.2.10.3.2-2.

**Table 7.2.10.3.2-1: URI query parameters supported by the PUT method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.10.3.2-2: Data structures supported by the PUT request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	DnsRule	1	The updated "state" is included in the payload body of the request.	
Response body	Data type	Cardinality	Response codes	Remarks
	DnsRule	1	200 OK	Upon success, a response body containing data type describing the updated DnsRule is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	412 Precondition Failed	It is used when a condition has failed during conditional requests, e.g. when using ETags to avoid write conflicts. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.

### 7.2.10.3.3 PATCH

Not supported.

### 7.2.10.3.4 POST

Not supported.

### 7.2.10.3.5 DELETE

Not supported.

## 7.2.11 Resource: confirm termination task

### 7.2.11.1 Description

This task resource allows a MEC application instance to confirm towards the MEC platform that it has completed the application level termination.

### 7.2.11.2 Resource definition

Resource URI: **{apiRoot}/mec\_app\_support/v1/applications/{appInstanceId}/confirm\_termination**

Resource URI variables for this resource are defined in table 7.2.11.2-1.

**Table 7.2.11.2-1: Resource URI variables for resource "confirm termination task"**

Name	Definition
apiRoot	See clause 7.2.2.
applInstanceid	Represents a MEC application instance.

### 7.2.11.3 Resource methods

#### 7.2.11.3.1 GET

Not supported.

#### 7.2.11.3.2 PUT

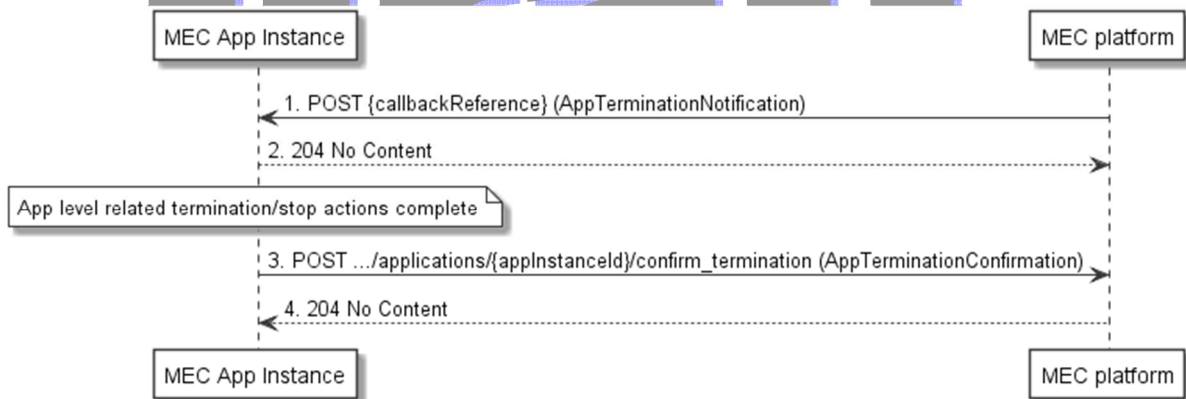
Not supported.

#### 7.2.11.3.3 PATCH

Not supported.

#### 7.2.11.3.4 POST

The high-level MEC application instance graceful termination/stop flow is introduced in clause 5.2.3, with the full detail provided in figure 7.2.11.3.4-1. In step 1 the MEC Platform notifies the MEC application instance that it is to be gracefully terminated/stopped. In step 2 the MEC application instance responds with a 204 No Content to acknowledge that it has received the terminate/stop notification. It can then execute application level terminate/stop related actions. In step 3, once such actions have been completed, the MEC application instance uses the POST method to confirm the application level termination of the MEC application instance. Finally, in step 4, the MEC Platform responds with a 204 No Content. This POST method shall support the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.11.3.4-1 and 7.2.11.3.4-2.

**Figure 7.2.11.3.4-1: MEC application termination/stop notification and confirmation using POST****Table 7.2.11.3.4-1: URI query parameters supported by the POST method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

Table 7.2.11.3.4-2: Data structures supported by the POST request/response on this resource

Request body	Data type	Cardinality	Remarks	
	AppTerminationConfirmation		Payload body in the request contains the operational action the application instance is responding to.	
Response body	Data type	Cardinality	Response codes	Remarks
	N/A		204 No Content	The request is acknowledged. The response body shall be empty.
	ProblemDetails	0..1	401 Unauthorized	It is used when the client did not submit the appropriate credentials. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	409 Conflict	The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is because the application instance resource is in NOT_INSTANTIATED state or because there is no termination ongoing. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	429 Too Many Requests	It is used when a rate limiter has triggered. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.

## 7.2.11.3.5

DELETE

Not supported.

## 7.2.12 Resource: confirm ready task

## 7.2.12.1 Description

This task resource allows a MEC application instance to confirm towards the MEC platform that it is up and running, which corresponds to step 4c described in clause 5.2.2.

## 7.2.12.2 Resource definition

Resource URI: `{apiRoot}/mec_app_support/v1/applications/{appInstanceId}/confirm_ready`

Resource URI variables for this resource are defined in table 7.2.12.2-1.

Table 7.2.12.2-1: Resource URI variables for resource "confirm ready task"

Name	Definition
apiRoot	See clause 7.2.2.
appInstanceId	Represents a MEC application instance.

## 7.2.12.3 Resource methods

## 7.2.12.3.1 GET

Not supported.

## 7.2.12.3.2 PUT

Not supported.

## 7.2.12.3.3 PATCH

Not supported.

## 7.2.12.3.4 POST

The POST method may be used by the MEC application instance to notify the MEC platform that it is up and running. POST HTTP method shall support the URI query parameters, request and response data structures, and response codes, as specified in tables 7.2.12.3.4-1 and 7.2.12.3.4-2.

**Table 7.2.12.3.4-1: URI query parameters supported by the POST method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 7.2.12.3.4-2: Data structures supported by the POST request/response on this resource**

Request body	Data type	Cardinality	Remarks	
AppReadyConfirmation		1	Payload body in the request contains the indication that the application instance is up and running.	
Response body	Data type	Cardinality	Response codes	Remarks
	N/A		204 No Content	The request is acknowledged. The response body shall be empty.
	ProblemDetails	0..1	401 Unauthorized	It is used when the client did not submit the appropriate credentials. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	409 Conflict	The operation cannot be executed currently, due to a conflict with the state of the resource. Typically, this is because the application instance resource is in NOT_INSTANTIATED state. The response body shall contain a ProblemDetails structure, in which the "detail" attribute shall convey more information about the error.
	ProblemDetails	0..1	429 Too Many Requests	It is used when a rate limiter has triggered. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.

## 7.2.12.3.5 DELETE

Not supported.

## 8 MEC service management API

### 8.1 Data model

#### 8.1.1 Introduction

Clauses 8.1.2 to 8.1.6 specify the data types that are used to implement the MEC service management API for which the relevant sequence diagrams are described in clause 5.2.

#### 8.1.2 Resource data types

##### 8.1.2.1 Introduction

This clause defines data structures to be used in resource representations.

##### 8.1.2.2 Type: ServiceInfo

This type represents the general information of a MEC service.

The attributes of the ServiceInfo shall follow the indications provided in table 8.1.2.2-1.

**Table 8.1.2.2-1: Attributes of ServiceInfo**

Attribute name	Data type	Cardinality	Description
serInstanceld	SerInstanceld	0..1	Identifier of the service instance assigned by the MEPM/MEC platform. For the uniqueness of the identifier across the MEC system, UUID format [i.7] is recommended. Shall be absent in POST requests, and present otherwise.
serName	SerName	1	The name of the service. This is how the service producing MEC application identifies the service instance it produces.
serCategory	CategoryRef	0..1	A Category reference. (The category resource is used to group product offerings, service and resource candidates in logical containers. Categories may contain other categories and/or product offerings, resource or service candidates.) (see note 1) For the serCategory, the example values include: 1. "RNI" 2. "Location" 3. "Bandwidth Management".
version	String	1	The version of the service.
state	ServiceState	1	Contains the service state.
transportId	String	0..1	Identifier of the platform-provided transport to be used by the service. Valid identifiers may be obtained using the "Transport information query" procedure. May be present in POST requests to signal the use of a platform-provided transport for the service, and shall be absent otherwise. See note 2.
transportInfo	TransportInfo	0..1	Information regarding the transport used by the service. May be present in POST requests to signal the use of an application-provided transport for the service, and shall be present otherwise. See note 2.
serializer	SerializerType	1	Indicate the supported serialization format of the service.

Attribute name	Data type	Cardinality	Description
scopeOfLocality	LocalityType	0..1	The scope of locality as expressed by "consumedLocalOnly" and "isLocal". If absent, defaults to MEC_HOST. See notes 3, 5 and 6.
consumedLocalOnly	Boolean	0..1	Indicate whether the service can only be consumed by the MEC applications located in the same locality (as defined by scopeOfLocality) as this service instance (TRUE) or not (FALSE). Default to TRUE if absent.
isLocal	Boolean	0..1	Indicate whether the service is located in the same locality (as defined by scopeOfLocality) as the consuming MEC application (TRUE) or not (FALSE). Default to TRUE if absent. See note 4.
livenessInterval	Integer	0..1	Interval (in seconds) between two consecutive "heartbeat" messages (see clause 8.2.10.3.3). If the service-producing application supports sending "heartbeat" messages, it shall include this attribute in the registration request. In this case, the application shall either set the value of this attribute to zero or shall use this attribute to propose a non-zero positive value for the liveness interval. If the application has provided this attribute in the request and the MEC platform requires "heartbeat" messages, the MEC platform shall return this attribute value in the HTTP responses. The MEC platform may use the value proposed in the request or may choose a different value. If the MEC platform does not require "heartbeat" messages for this service instance it shall omit the attribute in responses.
_links	Structure (inlined)	1	Links to resources related to this resource. Shall be absent in HTTP requests.
>self	LinkType	1	Link to this resource. Shall be present in HTTP responses.
>liveness	LinkType	0..1	Link to the "Individual mecServiceLiveness" resource where the MEC platform expects the service instance to send the liveness information. The structure of the URI of that resource is outside the scope of the present document. Shall be present in HTTP responses if the MEC platform requires "heartbeat" messages for this service instance and shall be absent otherwise.
NOTE 1: The service category may be included in the application descriptor. It may be allocated by the operator or by the application developer.			
NOTE 2: Either transportId or transportInfo but not both shall be present in POST requests.			
NOTE 3: Values NFVI_POP, ZONE and NFVI_NODE are used when the service instance is deployed as a VNF.			
NOTE 4: The isLocal is used only in service availability query response and service availability subscription/notification messages.			
NOTE 5: Value ZONE_GROUP can be used when the service instance is deployed as a VNF.			
NOTE 6: Regarding the value MEC_SYSTEM, if the service is running on the same MEC system as the MEC app, then it will be local to it.			

NOTE: In the present document it is not specified on service availability announcements outside a MEC system.

### 8.1.2.3 Type: TransportInfo

This type represents the transport information. The attributes of the TransportInfo type shall follow the indications provided in table 8.1.2.3-1.

Table 8.1.2.3-1: Attributes of TransportInfo

Attribute name	Data type	Cardinality	Description
id	String	1	The identifier of this transport.
name	String	1	The name of this transport.
description	String	0..1	Human-readable description of this transport.
type	TransportType	1	Type of the transport.
protocol	String	1	The name of the protocol used. Shall be set to "HTTP" for a REST API.
version	String	1	The version of the protocol used.
endpoint	EndPointInfo	1	Information about the endpoint to access the transport.
security	SecurityInfo	1	Information about the security used by the transport.
implSpecificInfo	Not specified	0..1	Additional implementation specific details of the transport.

#### 8.1.2.4 Type: ServiceLivenessInfo

This type represents the liveness information of a MEC service instance. The attributes of the "ServiceLivenessInfo" type shall follow the indications provided in table 8.1.2.4-1.

Table 8.1.2.4-1: Attributes of ServiceLivenessInfo

Attribute name	Data type	Cardinality	Description
state	ServiceState	1	Liveness state of the MEC service instance. The valid values are defined in clause 8.1.6.6.
timeStamp	Structure (inlined)	1	The time when the last "heartbeat" message was received by MEC platform.
>seconds	Uint32	1	The seconds part of the time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC.
>nanoSeconds	Uint32	1	The nanoseconds part of the time. Time is defined as Unix-time since January 1, 1970, 00:00:00 UTC.
interval	Integer	1	The interval (in seconds) between two consecutive "heartbeat" messages (see clause 8.2.10.3.3) that MEC platform has determined.

#### 8.1.2.5 Type: ServiceLivenessUpdate

This type represents changes to the liveness information of a MEC service instance, following the syntax of JSON Merge Patch specified in IETF RFC 7386 [17]. The "ServiceLivenessUpdate" type contains the subset of the attributes of the "ServiceLivenessInfo" type which are allowed to be modified by the PATCH method.

The attributes of the "ServiceLivenessUpdate" type shall follow the indications provided in table 8.1.2.5-1.

Table 8.1.2.5-1: Attributes of ServiceLivenessUpdate

Attribute name	Data type	Cardinality	Description
state	ServiceState	1	Update to the state of the MEC service instance to indicate that the service is still alive ("heartbeat" message). Shall be set to "ACTIVE".

### 8.1.3 Subscription data types

#### 8.1.3.1 Introduction

This clause defines data structures that define criteria to be used in subscriptions.

### 8.1.3.2 Type: SerAvailabilityNotificationSubscription

This type represents a subscription to the notifications from the MEC platform regarding the availability of a MEC service or a list of MEC services.

The attributes of the SerAvailabilityNotificationSubscription shall follow the indications provided in table 8.1.3.2-1.

**Table 8.1.3.2-1: Attributes of SerAvailabilityNotificationSubscription**

Attribute name	Data type	Cardinality	Description
subscriptionType	String	1	Shall be set to "SerAvailabilityNotificationSubscription".
callbackReference	Uri	1	URI selected by the MEC application instance to receive notifications on the subscribed MEC service availability information. This shall be included in both the request and the response.
_links	Structure (inlined)	0..1	Object containing hyperlinks related to the resource. This shall only be included in the HTTP responses.
>self	LinkType	1	Self-referring URI.
filteringCriteria	Structure (inlined)	0..1	Filtering criteria to match services for which events are requested to be reported. If absent, matches all services. All child attributes are combined with the logical "AND" operation.
>serInstanceIds	SerInstanceId	0..N	Identifiers of service instances about which to report events. See note.
>serNames	SerName	0..N	Names of services about which to report events. See note.
>serCategories	CategoryRef	0..N	Categories of services about which to report events. See note.
>states	ServiceState	0..N	States of the services about which to report events. If the event is a state change, this filter represents the state after the change.
>isLocal	Boolean	0..1	Restrict event reporting to whether the service is local to the MEC platform where the subscription is managed.
NOTE:	The attributes "serInstanceIds", "serNames" and "serCategories" provide mutually-exclusive alternatives to define a set of services. Only one of them may be present.		

## 8.1.4 Notification data types

### 8.1.4.1 Introduction

This clause defines data structures that define notifications.

### 8.1.4.2 Type: ServiceAvailabilityNotification

This type represents the service availability information that is used in the following cases:

- when the MEC platform announces the newly available services to the authorized relevant MEC applications (e.g. the applications that indicate the services as "optional" or "required") that are subscribed to the corresponding service availability notifications;
- when the MEC platform notifies the authorized relevant applications that are subscribed to the corresponding service availability notifications about the service availability changes.

The attributes of the ServiceAvailabilityNotification shall follow the indications provided in table 8.1.4.2-1.



Table 8.1.4.2-1: Attributes of ServiceAvailabilityNotification

Attribute name	Data type	Cardinality	Description
notificationType	String	1	Shall be set to "SerAvailabilityNotification".
serviceReferences	Structure (inlined)	1..N	List of links to services whose availability has changed.
>link	LinkType	0..1	Link to the resource representing the individual service. Shall be present unless "changeType"="REMOVED".
>serName	SerName	1	Name of the service
>serInstancelid	SerInstancelid	1	Identifier of the service
>state	ServiceState	1	State of the service after the modification.
>changeType	Enum (inlined)	1	Type of the change. Valid values: 1. ADDED: The service was newly added. 2. REMOVED: The service was removed. 3. STATE_CHANGED: Only the state of the service was changed. 4. ATTRIBUTES_CHANGED: At least one attribute of the service other than state was changed. The change may or may not include changing the state.
_links	Structure (inlined)	1	Object containing hyperlinks related to the resource.
>subscription	LinkType	1	A link to the related subscription.

## 8.1.5 Referenced structured data types

### 8.1.5.1 Introduction

This clause defines data structures that may be referenced from data structures defined in clauses 8.1.2 to 8.1.4, but may neither be resource representations nor notifications.

### 8.1.5.2 Type: CategoryRef

This type represents the category reference.

The attributes of the CategoryRef shall follow the indications provided in table 8.1.5.2-1.

Table 8.1.5.2-1: Attributes of CategoryRef

Attribute name	Data type	Cardinality	Description
href	Uri	1	Reference of the catalogue.
id	String	1	Unique identifier of the category.
name	String	1	Name of the category.
version	String	1	Category version.

### 8.1.5.3 Type: EndPointInfo

This type represents information about a transport endpoint. The attributes of the EndPointInfo shall follow the indications provided in table 8.1.5.3-1.

Table 8.1.5.3-1: Attributes of EndPointInfo

Attribute name	Data type	Cardinality	Description
uris	String	0..N	Entry point information of the service as string, formatted according to URI syntax (see IETF RFC 3986 [8]). Shall be used for REST APIs. See note.
fqdn	String	0..N	Fully Qualified Domain Name of the service. See note.
addresses	Structure (inlined)	0..N	Entry point information of the service as one or more pairs of IP address and port. See note.
>host	String	1	Host portion of the address.
>port	Int	1	Port portion of the address.
alternative	Not specified	0..1	Entry point information of the service in a format defined by an implementation, or in an external specification. See note.
NOTE: Exactly one of "uris", "fqdn", "addresses" or "alternative" shall be present.			

#### 8.1.5.4 Type: SecurityInfo

This type represents security information related to a transport.

In the present document, only security information for the client credentials grant type of OAuth 2.0 is specified. All parameters related to OAuth 2.0, including additional attributes that might need to be added when more grant types are supported in the future, shall be contained in the "oAuth2Info" structure. For the support of the OAuth 2.0 client credentials grant type, the attributes of the "oAuth2Info" attribute of the SecurityInfo shall follow the indications provided in table 8.1.5.4-1.

NOTE: For the use of alternative transport mechanisms by implementations, or for their specification in future versions of the present document, it is foreseen that the "SecurityInfo" structure may contain additional attributes that allow the MEC application to discover the applicable security-related parameters of these mechanisms.

Table 8.1.5.4-1: Attributes of SecurityInfo

Attribute name	Data type	Cardinality	Description
oAuth2Info	Structure (inlined)	0..1	Parameters related to use of OAuth 2.0. Shall be present in case OAuth 2.0 (see IETF RFC 6749 [13]) is supported to secure the provision of the service over the transport.
>grantTypes	Enum (inlined)	1..4	List of supported OAuth 2.0 grant types. Each entry shall be one of the following permitted values: <ul style="list-style-type: none"> <li>• OAUTH2_AUTHORIZATION_CODE (Authorization code grant type)</li> <li>• OAUTH2_IMPLICIT_GRANT (Implicit grant type)</li> <li>• OAUTH2_RESOURCE_OWNER (Resource owner password credentials grant type)</li> <li>• OAUTH2_CLIENT_CREDENTIALS (Client credentials grant type)</li> </ul> Only the value "OAUTH2_CLIENT_CREDENTIALS" is supported in the present document.
>tokenEndpoint	Uri	0..1	The token endpoint. Shall be present unless the grant type is OAUTH2_IMPLICIT_GRANT.
(extensions)	Not specified	0..N	Extensions for alternative transport mechanisms. These extensions depend on the actual transport, and are out of scope of the present document. For instance, such extensions may be used to signal the necessary parameters for the client to use TLS-based authorization defined for alternative transports (see ETSI GS MEC 009 [5] for more information).

## 8.1.6 Referenced simple data types and enumerations

### 8.1.6.1 Introduction

This clause defines simple data types and enumerations that can be referenced from data structures defined in clauses 8.1.2 to 8.1.5.

### 8.1.6.2 Simple data types

The simple data type defined for this API are provided in table 8.1.6.2-1.

**Table 8.1.6.2-1: Simple data types**

Type name	Description
SerInstanceld	String representing the identifier of the service
SerName	String representing the name of the service

### 8.1.6.3 Enumeration: SerializerType

The enumeration SerializerType represents types of serializers. This enumeration shall be extensible. It shall comply with the provisions defined in table 8.1.6.3-1.

**Table 8.1.6.3-1: Enumeration SerializerType**

Enumeration value	Description
JSON	Javascript object notation [9]
XML	eXtensible Mark-up Language version 1.1 [10]
PROTOBUF3	Protocol buffers version 3 [i.3]
NOTE:	The enumeration values above shall represent the serializers as defined by the referenced specifications.

### 8.1.6.4 Enumeration: TransportType

The enumeration TransportType represents types of transports. It shall comply with the provisions defined in table 8.1.6.4-1. This enumeration shall be extensible.

**Table 8.1.6.4-1: Enumeration TransportType**

Enumeration value	Description
REST_HTTP	RESTful API using HTTP (as defined in IETF RFC 7230 [11] and related specifications).
MB_TOPIC_BASED	Topic-based message bus which routes messages to receivers based on subscriptions, if a pattern passed on subscription matches the topic of the message. EXAMPLE: MQTT (see [i.4]).
MB_ROUTING	Routing-based message bus which routes messages to receivers based on subscriptions, if a key passed on subscription is equal to the key of the message.
MB_PUBSUB	Publish-subscribe based message bus which distributes messages to all subscribers.
RPC	Remote procedure call. EXAMPLE: GRPC (see [i.5]).
RPC_STREAMING	Remote procedure call supporting streams of requests and responses. EXAMPLE: GRPC (see [i.5]).
WEBSOCKET	Websockets as defined in IETF RFC 6455 [12].

### 8.1.6.5 Enumeration: LocalityType

The enumeration LocalityType represents types of locality. It shall comply with the provisions defined in table 8.1.6.5-1.

**Table 8.1.6.5-1: Enumeration LocalityType**

Enumeration value	Description
MEC_SYSTEM	MEC system
MEC_HOST	MEC host
NFVI_POP	NFVI PoP
ZONE	Resource zone, as defined in ETSI GS NFV-IFA 007 [15]
ZONE_GROUP	Group of resource zones, as defined in ETSI GS NFV-IFA 007 [15]
NFVI_NODE	NFVI node

NOTE: In the present document it is not specified on service availability announcements outside a MEC system.

### 8.1.6.6 Enumeration: ServiceState

The enumeration ServiceState represents possible states of a MEC service instance. This enumeration shall comply with the provisions defined in table 8.1.6.6-1.

**Table 8.1.6.6-1: Enumeration ServiceState**

Enumeration value	Description
ACTIVE	The service is active.
INACTIVE	The service is inactive.
SUSPENDED	The service is suspended because its producer did not send a "heartbeat" message in the expected time interval.

## 8.2 API definition

### 8.2.1 Introduction

This clause defines the resources and operations of the MEC service management API.

### 8.2.2 Global definitions and resource structure

All resource URIs of this API shall have the following root:

- **{apiRoot}/{apiName}/{apiVersion}/**

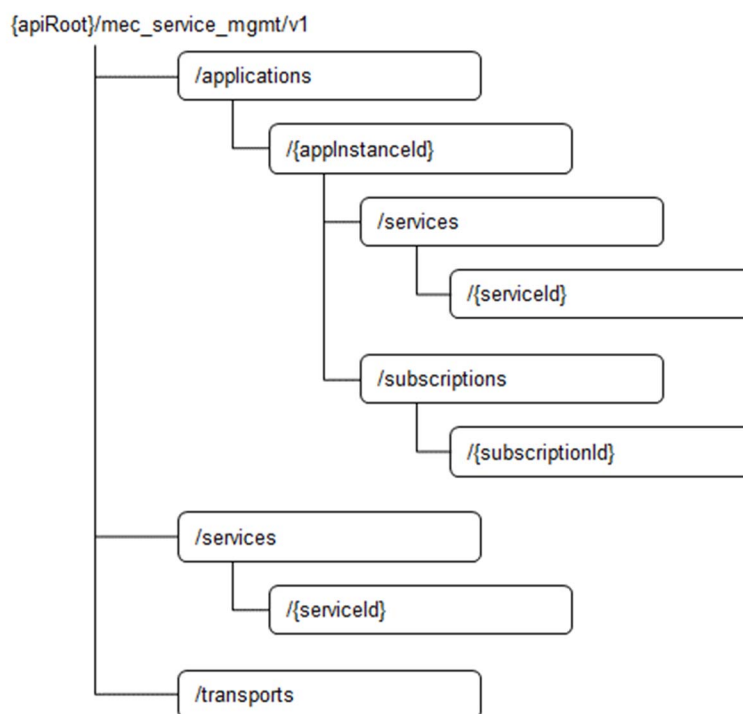
The "apiRoot" is discovered using the service registry. The "apiName" shall be set to "mec\_service\_mgmt" and the "apiVersion" shall be set to "v1" for the present document. It includes the scheme ("https"), host and optional port, and an optional prefix string. All resource URIs in clauses 8.2.3 to 8.2.10 are defined relative to the above root URI.

The API shall support HTTP over TLS (also known as HTTPS) using TLS version 1.2 (as defined by IETF RFC 5246 [7]). TLS 1.3 (including the new specific requirements for TLS 1.2 implementations) defined by IETF RFC 8446 [18] should be supported. HTTP without TLS shall not be used. Versions of TLS earlier than 1.2 shall neither be supported nor used.

This API shall require the use of the OAuth 2.0 client credentials grant type according to IETF RFC 6749 [13] with bearer tokens according to IETF RFC 6750 [14]. See clause 6.16 of ETSI GS MEC 009 [5] for more information. How the token endpoint and client credentials are provisioned into the MEC applications is out of scope of the present document.

This API supports additional application-related error information to be provided in the HTTP response when an error occurs. See clause 6.15 of ETSI GS MEC 009 [5] for more information.

Figure 8.2.2-1 illustrates the resource URI structure of this API.



**Figure 8.2.2-1: Resource URI structure of the MEC service management API**

Table 8.2.2-1 provides an overview of the resources defined by the present specification for the MEC applications support API, and the applicable HTTP methods.

**Table 8.2.2-1: Resources and methods overview**

Resource name	Resource URI	HTTP method	Meaning
A list of mecService	/services	GET	Retrieve information about a list of mecService resources
Individual mecService	/services/{serviceId}	GET	Retrieve information about a mecService resource
A list of mecService of an application instance	/applications/{applInstancelD}/services	GET	Retrieve information about a list of mecService resources of an application instance
		POST	Create a mecService resource of an application instance
Individual mecService of an application instance	/applications/{applInstancelD}/services/{serviceId}	GET	Retrieve information about a mecService resource of an application instance
		PUT	Update the information about a mecService resource of an application instance
		DELETE	Delete a mecService resource
Parent resource of all mecSrvMgmtSubscription of a subscriber	/applications/{applInstancelD}/subscriptions	GET	Retrieve information about a list of mecSrvMgmtSubscription resources for this subscriber
		POST	Create a mecSrvMgmtSubscription resource
Individual mecSrvMgmtSubscription	/applications/{applInstancelD}/subscriptions/{subscriptionId}	GET	Retrieve information about a mecSrvMgmtSubscription resource for this subscriber
		DELETE	Delete a mecSrvMgmtSubscription resource
A list of mecTransport	/transports	GET	Retrieve information about the available transports

Resource name	Resource URI	HTTP method	Meaning
Individual mecServiceLiveness	See note.	GET	Retrieve information about the liveness of a MEC service instance produced by an application instance.
		PATCH	Send a "heartbeat" message related to a MEC service instance.
NOTE: The URI of this resource is allocated by the MEC platform.			

## 8.2.3 Resource: a list of mecService

### 8.2.3.1 Description

This resource is used to represent a list of MEC service instances.

### 8.2.3.2 Resource definition

Resource URI: {apiRoot}/mec\_service\_mgmt/v1/services

Resource URI variables for this resource are defined in table 8.2.3.2-1.

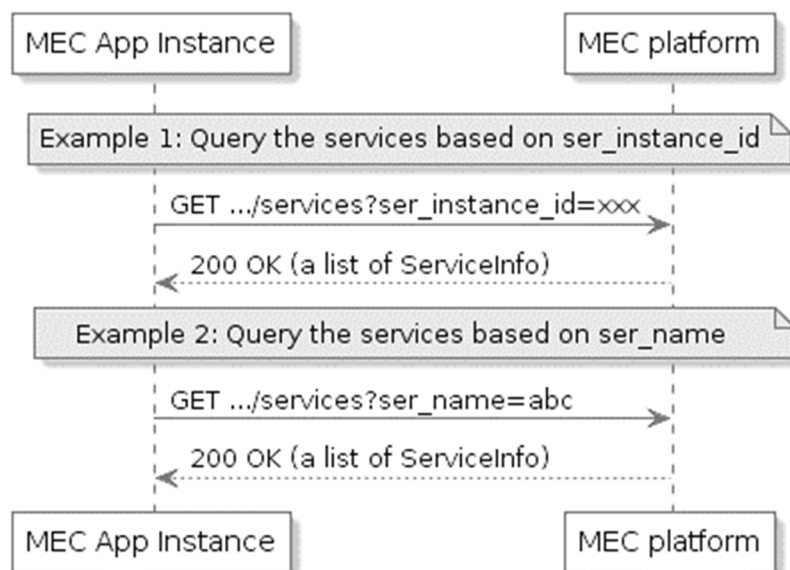
**Table 8.2.3.2-1: Resource URI variables for resource "a list of mecService"**

Name	Definition
apiRoot	See clause 8.2.2

### 8.2.3.3 Resource methods

#### 8.2.3.3.1 GET

This method retrieves information about a list of mecService resources. This method is typically used in "service availability query" procedure as described in clause 5.2.5. Figure 8.2.3.3.1-1 shows the example message flows using GET method.



**Figure 8.2.3.3.1-1: Service availability query**

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.3.3.1-1 and 8.2.3.3.1-2. When no URI query parameter is present, all the relevant mecService resources to the requestor will be returned.

**Table 8.2.3.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
ser_instance_id	String	0..N	A MEC application instance may use multiple ser_instance_ids as an input parameter to query the availability of a list of MEC service instances. See note.
ser_name	String	0..N	A MEC application instance may use multiple ser_names as an input parameter to query the availability of a list of MEC service instances. See note.
ser_category_id	String	0..1	A MEC application instance may use ser_category_id as an input parameter to query the availability of a list of MEC service instances in a serCategory. See note.
scope_of_locality	LocalityType	0..1	A MEC application instance may use scope_of_locality as an input parameter to query the availability of a list of MEC service instances with a certain scope of locality, as defined in LocalityType in table 8.1.6.5-1.
consumed_local_only	Boolean	0..1	A MEC application instance may use consumed_local_only as an input parameter to query the availability of a list of MEC service instances that can be consumed only locally.
is_local	Boolean	0..1	A MEC application instance may use is_local as an input parameter to query the availability of a list of MEC service instances in the local MEC host or in local and remote MEC hosts.

NOTE: Either "ser\_instance\_id" or "ser\_name" or "ser\_category\_id" or none of them shall be present.

**Table 8.2.3.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	0..N	200 OK	Upon success, a response body containing an array of the mecServices is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	414 URI Too Long	It is used to indicate that the server is refusing to process the request because the request URI is longer than the server is willing or able to process.

### 8.2.3.3.2 PUT

Not supported.

### 8.2.3.3.3 PATCH

Not supported.

## 8.2.3.3.4 POST

Not supported.

## 8.2.3.3.5 DELETE

Not supported.

## 8.2.4 Resource: individual mecService

## 8.2.4.1 Description

This resource is used to represent a MEC service instance, which follows the resource data type of "ServiceInfo" as specified in clause 8.1.2.2.

## 8.2.4.2 Resource definition

Resource URI: {apiRoot}/mec\_service\_mgmt/v1/services/{serviceId}

Resource URI variables for this resource are defined in table 8.2.4.2-1.

**Table 8.2.4.2-1: Resource URI variables for resource "individual mecService"**

Name	Definition
apiRoot	See clause 8.2.2
serviceId	Represents a MEC service instance

## 8.2.4.3 Resource methods

## 8.2.4.3.1 GET

This method retrieves information about a mecService resource. This method is typically used in "service availability query" procedure as described in clause 5.2.5.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.4.3.1-1 and 8.2.4.3.1-2.

**Table 8.2.4.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			



Table 8.2.4.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	1	200 OK	It is used to indicate nonspecific success. The response body contains a representation of the resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 8.2.4.3.2 PUT

Not supported.

## 8.2.4.3.3 PATCH

Not supported.

## 8.2.4.3.4 POST

Not supported.

## 8.2.4.3.5 DELETE

Not supported.

Draft

## 8.2.5 Resource: a list of mecTransport

## 8.2.5.1 Description

This resource is used to represent a list of transports provided by the MEC platform.

## 8.2.5.2 Resource definition

Resource URI: {apiRoot}/mec\_service\_mgmt/v1/transports

Resource URI variables for this resource are defined in table 8.2.5.2-1.

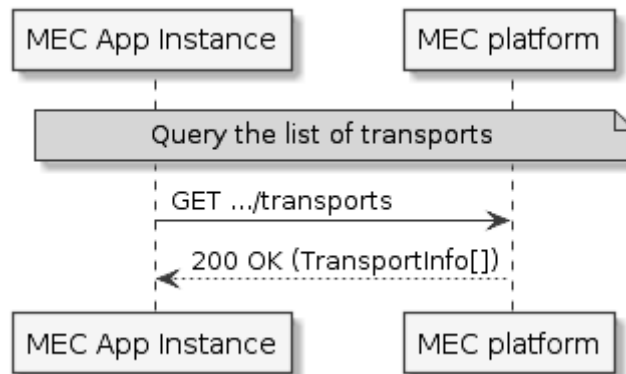
Table 8.2.5.2-1: Resource URI variables for resource "a list of mecTransport"

Name	Definition
apiRoot	See clause 8.2.2.

### 8.2.5.3 Resource methods

#### 8.2.5.3.1 GET

This method retrieves information about a list of available transports. This method is typically used by a service-producing application to discover transports provided by the MEC platform in the "transport information query" procedure as described in clause 5.2.9. Figure 8.2.5.3.1-1 shows the example message flows using GET method.



**Figure 8.2.5.3.1-1: Transport information query**

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.5.3.1-1 and 8.2.5.3.1-2.

**Table 8.2.5.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.5.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	TransportInfo	0..N	200 OK	Upon success, a response body containing an array describing the available transports is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden		The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 8.2.5.3.2 PUT

Not supported.

## 8.2.5.3.3 PATCH

Not supported.

## 8.2.5.3.4 POST

Not supported.

## 8.2.5.3.5 DELETE

Not supported.

## 8.2.6 Resource: a list of mecService of an application instance

## 8.2.6.1 Description

This resource is used to represent a list of MEC service instances that is associated with an application instance.

## 8.2.6.2 Resource definition

Resource URI: **{apiRoot}/mec\_service\_mgmt/v1/applications/{appInstanceId}/services**

Resource URI variables for this resource are defined in table 8.2.6.2-1.

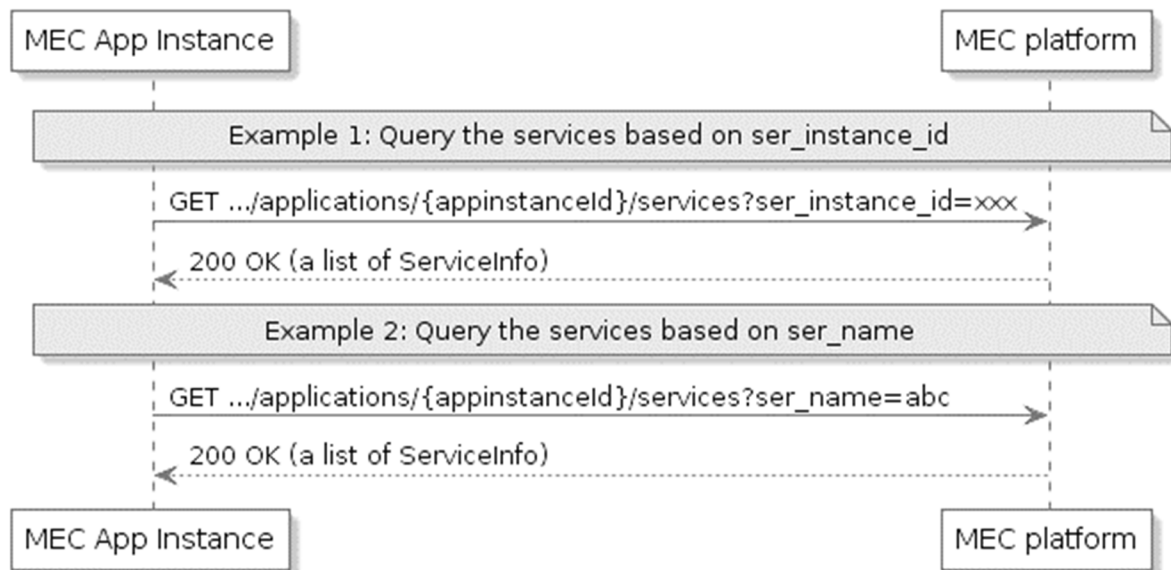
**Table 8.2.6.2-1: Resource URI variables for resource "a list of mecService of an application instance"**

Name	Definition
apiRoot	See clause 8.2.2.
appInstanceId	Represents a MEC application instance.

## 8.2.6.3 Resource methods

## 8.2.6.3.1 GET

This method retrieves information about a list of mecService resources that is associated with an application instance. This method is typically used in "service availability query" procedure as described in clause 5.2.5. Figure 8.2.6.3.1-1 shows the example message flows using GET method.



**Figure 8.2.6.3.1-1: Service availability query**

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.6.3.1-1 and 8.2.6.3.1-2. When no URI query parameter is present, all the relevant mecService resources to the requestor will be returned.

**Table 8.2.6.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
ser_instance_id	String	0..N	A MEC application instance may use multiple ser_instance_ids as an input parameter to query the availability of a list of MEC service instances. See note.
ser_name	String	0..N	A MEC application instance may use multiple ser_names as an input parameter to query the availability of a list of MEC service instances. See note.
ser_category_id	String	0..1	A MEC application instance may use ser_category_id as an input parameter to query the availability of a list of MEC service instances in a serCategory. See note.
scope_of_locality	LocalityType	0..1	A MEC application instance may use scope_of_locality as an input parameter to query the availability of a list of MEC service instances with a certain scope of locality, as defined in LocalityType in table 8.1.6.5-1.
consumed_local_only	Boolean	0..1	A MEC application instance may use consumed_local_only as an input parameter to query the availability of a list of MEC service instances that can be consumed only locally.
is_local	Boolean	0..1	A MEC application instance may use is_local as an input parameter to query the availability of a list of MEC service instances in the local MEC host or in local and remote MEC hosts.

NOTE: Either "ser\_instance\_id" or "ser\_name" or "ser\_category\_id" or none of them shall be present.

Table 8.2.6.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	0..N	200 OK	Upon success, a response body containing an array of the mecServices is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	0..1	414 URI Too Long	It is used to indicate that the server is refusing to process the request because the request URI is longer than the server is willing or able to process.

## 8.2.6.3.2 PUT

Not supported.

## 8.2.6.3.3 PATCH

Not supported.

## 8.2.6.3.4 POST

This method is used to create a mecService resource that is associated with the application instance. This method is typically used in "service availability update and new service registration" procedure as described in clause 5.2.4. Figure 8.2.6.3.4-1 shows the message flow.

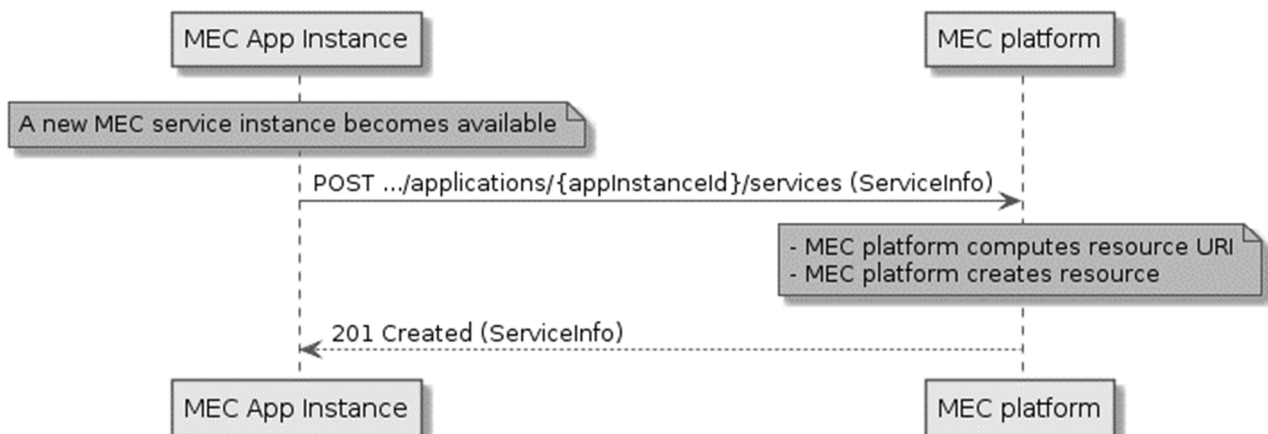


Figure 8.2.6.3.4-1: New service registration

POST HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.6.3.4-1 and 8.2.6.3.4-2.

Table 8.2.6.3.4-1: URI query parameters supported by the POST method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 8.2.6.3.4-2: Data structures supported by the POST request/response on this resource

Request body	Data type	Cardinality	Remarks	
	ServiceInfo	1	Payload body in the request contains ServiceInfo to be created.	
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	1	201 Created	Upon success, the HTTP response shall include a "Location" HTTP header that contains the resource URI of the created resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

### 8.2.6.3.5 DELETE

Not supported.

## 8.2.7 Resource: individual mecService of an application instance

### 8.2.7.1 Description

This resource is used to represent a MEC service instance that is associated with an application instance, which follows the resource data type of "ServiceInfo" as specified in clause 8.1.2.2.

### 8.2.7.2 Resource definition

Resource URI: {apiRoot}/mec\_service\_mgmt/v1/applications/{appInstanceId}/services/{serviceId}

Resource URI variables for this resource are defined in table 8.2.7.2-1.

Table 8.2.7.2-1: Resource URI variables for resource "individual mecService of an application instance"

Name	Definition
apiRoot	See clause 8.2.2
appInstanceId	Represents a MEC application instance
serviceId	Represents a MEC service instance

### 8.2.7.3 Resource methods

#### 8.2.7.3.1 GET

This method retrieves information about a mecService resource that is associated with an application instance. This method is typically used in "service availability query" procedure as described in clause 5.2.5.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.7.3.1-1 and 8.2.7.3.1-2.

**Table 8.2.7.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.7.3.1-2: Data structures supported by the GET request/response on this resource**

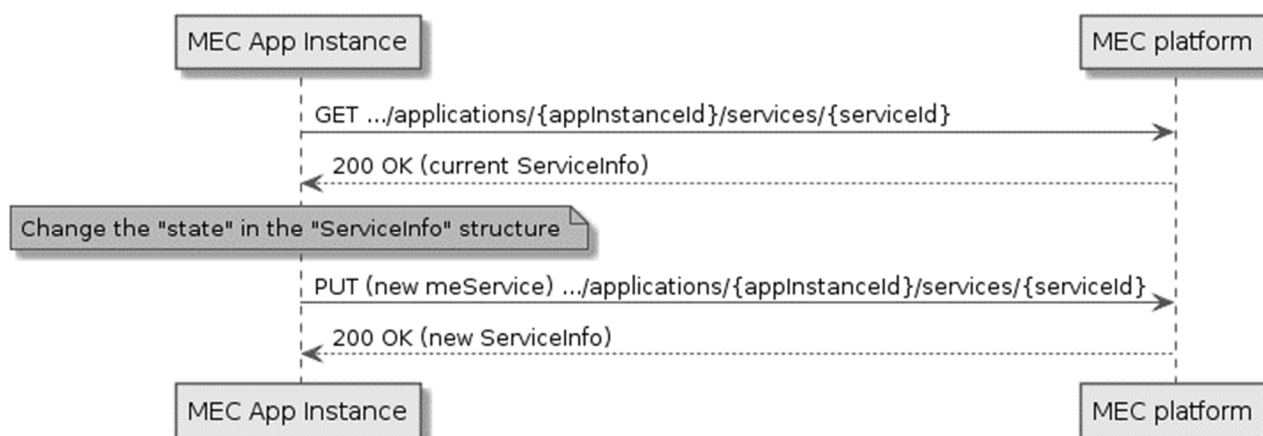
Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	1	200 OK	It is used to indicate nonspecific success. The response body contains a representation of the resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.	

### 8.2.7.3.2

#### PUT

This method updates the information about a mecService resource that is associated with the application instance. As specified in ETSI GS MEC 009 [5], the PUT HTTP method has "replace" semantics.

PUT method is typically used in "service availability update" procedure as described in clause 5.2.4. Figure 8.2.7.3.2-1 shows the message flow using PUT.



**Figure 8.2.7.3.2-1: Service availability update using PUT**

PUT HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.7.3.2-1 and 8.2.7.3.2-2.

Table 8.2.7.3.2-1: URI query parameters supported by the PUT method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 8.2.7.3.2-2: Data structures supported by the PUT request/response on this resource

Request body	Data type	Cardinality	Remarks	
	ServiceInfo	1	One or more updated attributes that are allowed to be changed (i.e. "state" or other attributes based on definition in clause 8.1.2.2) are included in the ServiceInfo data structure in the payload body of the request.	
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceInfo	1	200 OK	Upon success, a response body containing data type describing the updated ServiceInfo is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
ProblemDetails	0..1	412 Precondition Failed	It is used when a condition has failed during conditional requests, e.g. when using ETags to avoid write conflicts. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.	

## 8.2.7.3.3

## PATCH

Not supported.

## 8.2.7.3.4

## POST

Not supported.

## 8.2.7.3.5

## DELETE

This method deletes a mecService resource. This method is typically used in the service deregistration procedure. Figure 8.2.7.3.5-1 shows the example message flows using DELETE method.

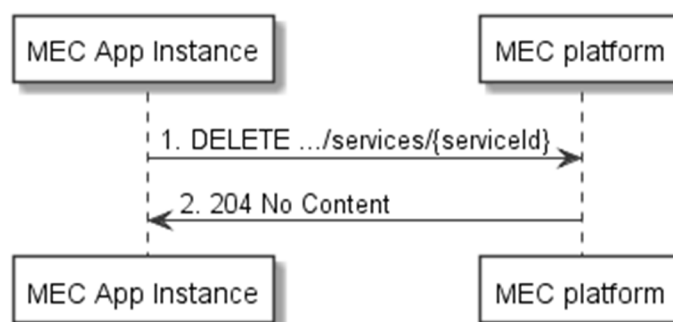


Figure 8.2.7.3.5-1: Service deregistration



DELETE HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.7.3.5-1 and 8.2.7.3.5-2.

**Table 8.2.7.3.5-1: URI query parameters supported by the DELETE method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.7.3.5-2: Data structures supported by the DELETE request on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	n/a		204 No Content	The operation has been successful. The response body shall be empty.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 8.2.8 Resource: all mecSrvMgmtSubscription

### 8.2.8.1 Description

This resource is used to represent all subscriptions of a subscriber to the notifications from the MEC platform.

### 8.2.8.2 Resource definition

Resource URI: `{apiRoot}/mec_service_mgmt/v1/applications/{appInstanceId}/subscriptions`

Resource URI variables for this resource are defined in table 8.2.8.2-1.

**Table 8.2.8.2-1: Resource URI variables for resource "all mecSrvMgmtSubscription"**

Name	Definition
apiRoot	See clause 8.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.

### 8.2.8.3 Resource methods

#### 8.2.8.3.1 GET

The GET method may be used to request information about all subscriptions for this requestor. Upon success, the response contains payload body with all the subscriptions for the requestor.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.8.3.1-1 and 8.2.8.3.1-2.

Table 8.2.8.3.1-1: URI query parameters supported by the GET method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 8.2.8.3.1-2: Data structures supported by the GET request/response on this resource

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	SubscriptionLinkList	1	200 OK	Upon success, a response body containing the list of links to the requested subscriptions is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 8.2.8.3.2 PUT

Not supported.

## 8.2.8.3.3 PATCH

Not supported.

## 8.2.8.3.4 POST

The POST method may be used to create a new subscription. One example use case is to create a new subscription to the MEC service availability notifications. Upon success, the response contains payload body describing the created subscription. This method is typically used in "Subscribing to service availability event notifications" procedure as described in clause 5.2.6.2. Figure 8.2.8.3.4-1 shows the example message flows using POST method.



Figure 8.2.8.3.4-1: Subscribing to service availability event notifications

POST HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.8.3.4-1 and 8.2.8.3.4-2.

Table 8.2.8.3.4-1: URI query parameters supported by the POST method on this resource

Name	Data type	Cardinality	Remarks
n/a			

Table 8.2.8.3.4-2: Data structures supported by the POST request/response on this resource

Request body	Data type	Cardinality	Remarks	
	SerAvailabilityNotificationSubscription	1	Payload body in the request contains a subscription to the MEC service availability notifications that is to be created.	
Response body	Data type	Cardinality	Response codes	Remarks
	SerAvailabilityNotificationSubscription	1	201 Created	Upon success, the HTTP response shall include a "Location" HTTP header that contains the resource URI of the created subscription resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 8.2.8.3.5

## DELETE

Not supported.

## 8.2.9 Resource: individual mecSrvMgmtSubscription

## 8.2.9.1 Description

This resource is used to represent a subscription to the notifications from the MEC platform. When this resource represents a subscription to the notifications regarding the availability of a MEC service or a list of MEC services, it shall follow the data type of "SerAvailabilityNotificationSubscription" as specified in clause 8.1.3.2. The notifications that are related to a meSerAvailSubscription follow the data type of "ServiceAvailabilityNotification" as specified in clause 8.1.4.2.

## 8.2.9.2 Resource definition

Resource URI: {apiRoot}/mec\_service\_mgmt/v1/applications/{appInstanceId}/subscriptions/{subscriptionId}

Resource URI variables for this resource are defined in table 8.2.9.2-1.

Table 8.2.9.2-1: Resource URI variables for resource "individual mecSrvMgmtSubscription"

Name	Definition
apiRoot	See clause 8.2.2.
appInstanceId	Represents a MEC application instance. Note that the appInstanceId is allocated by the MEC platform manager.
subscriptionId	Represents a subscription to the notifications from the MEC platform.

### 8.2.9.3 Resource methods

#### 8.2.9.3.1 GET

The GET method requests information about a subscription for this requestor. Upon success, the response contains payload body with the subscription for the requestor.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.9.3.1-1 and 8.2.9.3.1-2.

**Table 8.2.9.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.9.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body			Response codes	Remarks
	SerAvailabilityNotificationSubscription	1	200 OK	Upon success, a response body containing the requested subscription is returned.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

#### 8.2.9.3.2 PUT

Not supported.

#### 8.2.9.3.3 PATCH

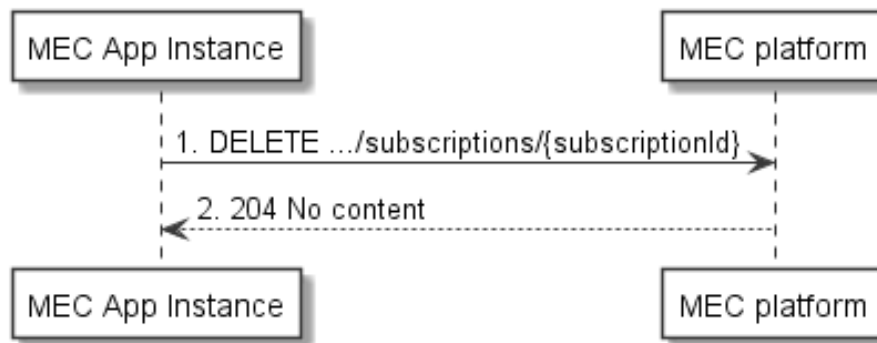
Not supported.

#### 8.2.9.3.4 POST

Not supported.

#### 8.2.9.3.5 DELETE

This method deletes a mecSrvMgmtSubscription. This method is typically used in "Unsubscribing from event notifications" procedure as described in clause 5.2.6.3. Figure 8.2.9.3.5-1 shows the example message flows using DELETE method.



**Figure 8.2.9.3.5-1: Unsubscribing from MEC service management event notifications**

DELETE HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.9.3.5-1 and 8.2.9.3.5-2.

**Table 8.2.9.3.5-1: URI query parameters supported by the DELETE method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.9.3.5-2: Data structures supported by the DELETE request on this resource**

Request body	Data type	Cardinality	Remarks	
	n/a			
Response body	Data type	Cardinality	Response codes	Remarks
	n/a		204 No Content	
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

## 8.2.10 Resource: individual mecServiceLiveness

### 8.2.10.1 Description

This resource is used to represent the liveness of a MEC service instance produced by an application instance.

### 8.2.10.2 Resource definition

Resource URI: **(allocated by the MEC platform)**

The URI of this resource is allocated by the MEC platform at the time of MEC service registration and signalled in the "liveness" link in the representation of the related "Individual mecService of an application instance" resource.

Resource URI variables for this resource are defined in table 8.2.10.2-1.

**Table 8.2.10.2-1: Resource URI variables for resource "individual mecServiceLiveness"**

Name	Definition
none specified	

### 8.2.10.3 Resource methods

#### 8.2.10.3.1 GET

This method retrieves information about an "Individual mecServiceLiveness" resource.

This method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.10.3.1-1 and 8.2.10.3.1-2.

**Table 8.2.10.3.1-1: URI query parameters supported by the GET method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

**Table 8.2.10.3.1-2: Data structures supported by the GET request/response on this resource**

Request body	Data type	Cardinality	Remarks	
n/a				
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceLivenessInfo	1	200 OK	It is used to indicate nonspecific success. The response body contains a representation of the resource.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.

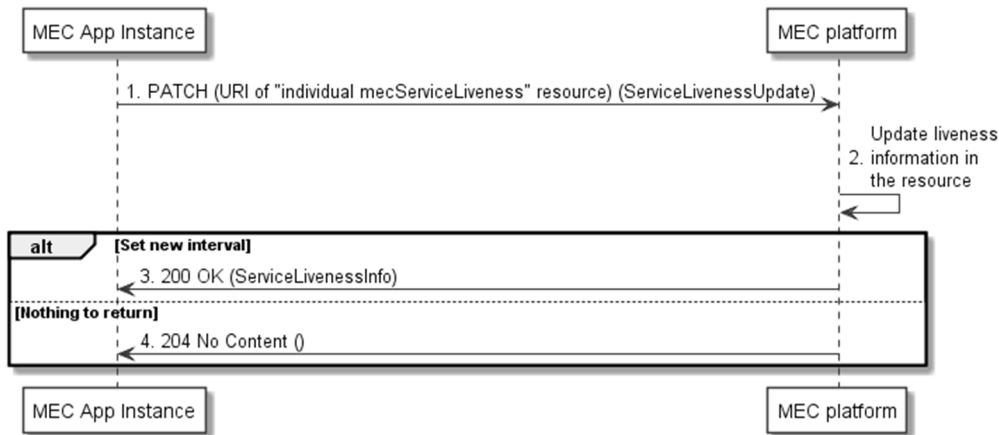
#### 8.2.10.3.2 PUT

Not supported.

#### 8.2.10.3.3 PATCH

As specified in ETSI GS MEC 009 [5], the PATCH HTTP method (see IETF RFC 5789 [16]) is used to update a resource on top of the existing resource state with partial changes described by the client. As opposed to PUT, PATCH does not carry a representation of the resource in the payload body, but a "deltas document" (see the definition of the "ServiceLivenessUpdate" type in clause 8.1.2.5) that instructs the server how to modify the resource representation.

The PATCH method is used in the "Service heartbeat" procedure as described in clause 5.2.12 and is referred to also as "heartbeat" message. Figure 8.2.10.3.3-1 shows the message flow using PATCH. It is the intent of this message to (re)confirm the "ACTIVE" state, but not to change the state from "INACTIVE" to "ACTIVE".



**Figure 8.2.10.3.3-1: Service liveness update using PATCH**

- 1) The MEC application instance that provides MEC service shall send a PATCH request to the resource URI representing the liveness of the service instance.
- 6) The MEC platform shall update the liveness resource as follows: It shall record the time when the message was received in the "timeStamp" attribute. Also, if the "state" attribute in the resource contains the value "SUSPENDED" and the "state" attribute in the payload body contains the value "ACTIVE", it shall set the value of the "state" attribute in the resource to that value.
- 7) If there is no payload body to return upon successful execution, the MEC platform shall return "204 No Content".
- 8) Alternatively, if the MEC platform intends to instruct the application to use a new liveness "interval" value for the service instance, it shall return "200 OK" along with the full ServiceLivenessInfo.

Error condition: Overwriting the "INACTIVE" state in a "heartbeat" message is forbidden and results in an error.

The PATCH HTTP method shall comply with the URI query parameters, request and response data structures, and response codes, as specified in tables 8.2.10.3.3-1 and 8.2.10.3.3-2.

**Table 8.2.10.3.3-1: URI query parameters supported by the PATCH method on this resource**

Name	Data type	Cardinality	Remarks
n/a			

Table 8.2.10.3.3-2: Data structures supported by the PATCH request/response on this resource

Request body	Data type	Cardinality	Remarks	
	ServiceLivenessUpdate	1	It contains an update of the liveness state.	
Response body	Data type	Cardinality	Response codes	Remarks
	ServiceLivenessInfo	1	200 OK	Upon success, a response body is returned containing the updated liveness interval value of the service Instance.
	n/a		204 No Content	Successful response sent when there is no need to provide a new liveness interval value to the service Instance.
	ProblemDetails	0..1	400 Bad Request	It is used to indicate that incorrect parameters were passed to the request. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	404 Not Found	It is used when a client provided a URI that cannot be mapped to a valid resource URI. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.
	ProblemDetails	0..1	403 Forbidden	The operation is not allowed given the current status of the resource. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
	ProblemDetails	1	409 Conflict	The operation is not allowed due to a conflict with the state of the resource. The MEC platform shall respond with this code if the service instance is in "INACTIVE" state. More information shall be provided in the "detail" attribute of the "ProblemDetails" structure.
ProblemDetails	0..1	412 Precondition Failed	It is used when a condition has failed during conditional requests, e.g. when using ETags to avoid write conflicts. In the returned ProblemDetails structure, the "detail" attribute should convey more information about the error.	

## 8.2.10.3.4 POST

Not supported.

## 8.2.10.3.5 DELETE

Not supported.



---

## Annex A (informative): Complementary material for API utilization

To complement the definitions for each method and resource defined in the interface clauses of the present document, ETSI ISG MEC is providing for each MEC Platform Application Enablement API a supplementary description file compliant to the OpenAPI Specification [i.6].

In case of discrepancies between each supplementary description file and the related data structure definitions in the present document, the data structure definitions take precedence.

The supplementary description files, relating to the present document, are located at <https://forge.etsi.org/rep/mec/gs011-app-enablement-api>.

**Draft**

# Annex B (informative): Mapping MEC service management API to 3GPP CAPIF APIs

## B.0 Definitions (ETSI TS 123 222)

**API invoker:** The entity which invokes the CAPIF or service APIs.

**API exposing function:** The entity which provides the service communication entry point for the service APIs.

**Common API framework:** A framework comprising common API aspects that are required to support service APIs.

**Northbound API:** A service API exposed to higher-layer API invokers.

## B.1 Introduction

In 3GPP, there are multiple northbound API-related specifications. To avoid duplication and inconsistency of approach between different API specifications, 3GPP has developed a common API framework (CAPIF) that includes common aspects applicable to any northbound service APIs. The common API framework applies to both EPS and 5GS, and is independent of the underlying 3GPP access (e.g. E-UTRA, NR).

ETSI TS 123 222 [i.8] specifies the architecture, procedures and information flows necessary for the CAPIF, while ETSI TS 129 222 [i.9] describes the protocol for the CAPIF for 3GPP Northbound APIs. The CAPIF functional model is organized into functional entities to describe a functional architecture which enables an API invoker to access and invoke service APIs. The relationship between the MEC API framework and the CAPIF is shown in figure B.1-1.

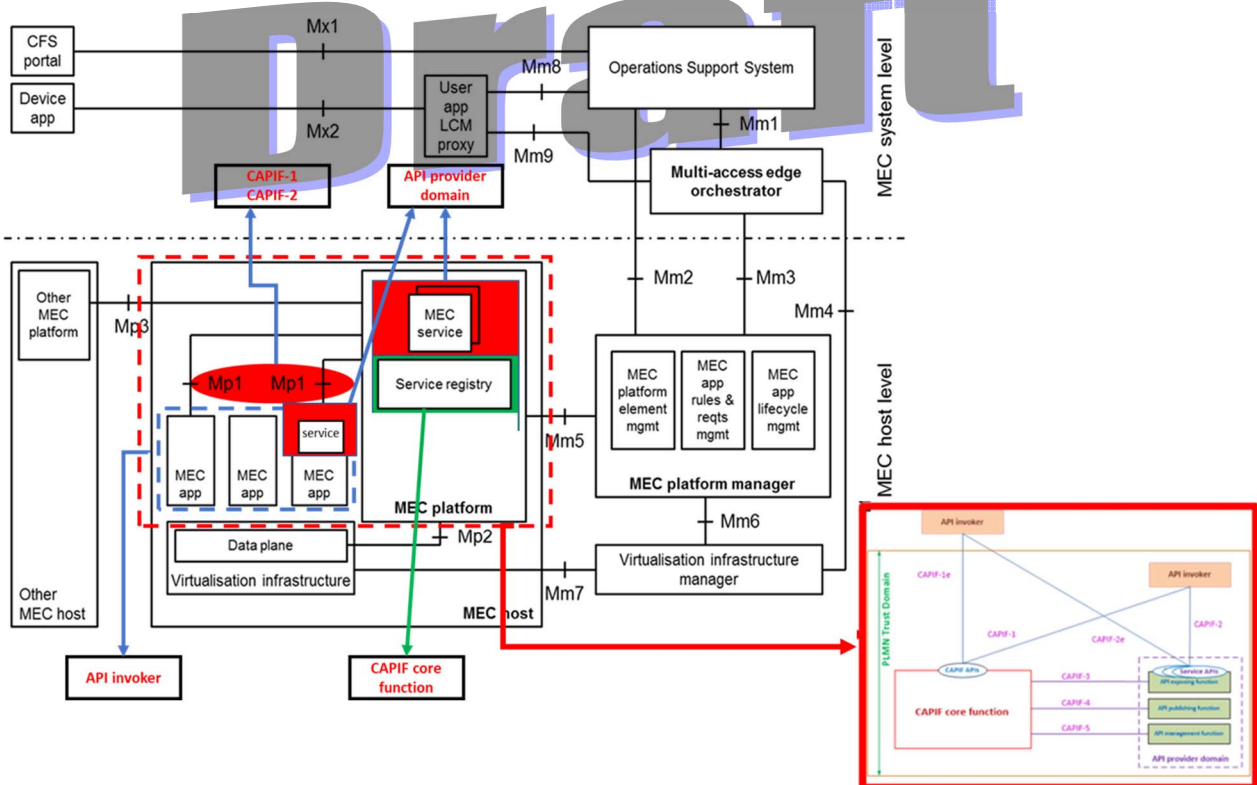


Figure B.1-1: Relationship between MEC and 3GPP CAPIF

MEC platform includes API-related functionality such as service registry, which is equivalent to the API registry of the CAPIF core function. Similarly the MEC platform can also expose MEC service APIs for consumption by MEC

applications. The existing MEC platform functionality related to API enablement, can be mapped into the CAPIF core function.

The API provider domain in CAPIF collectively represents the service APIs available for consumption in any 5G network functions and any trusted 3<sup>rd</sup> party application functions. A MEC service produced by a MEC application or the MEC platform can be mapped into the API provider domain in CAPIF. A MEC application or MEC platform consuming a service is an API invoker in CAPIF.

---

## B.2 Mapping MEC service management API to CAPIF APIs

### B.2.1 Overview

The MEC service management API (e.g. service registry, service discovery and service announcement) defined in the present document is similar to the corresponding CAPIF APIs (e.g. CAPIF\_Discover\_Service\_API, CAPIF\_Publish\_Service\_API and CAPIF\_Events\_API). Different ways of mapping are described in the following clauses to simplify the task for developers who need to interact with both. The mapping between the MEC and CAPIF APIs include:

- Mapping of the URI structures.
- Mapping of the service discovery query parameters.
- Mapping of the data models for the payload bodies of the RESTful protocols.

### B.2.2 Mapping of the resource structures

Table B.2.2-1 shows the mapping of MEC service management API resources defined in the present document to CAPIF resources defined in ETSI TS 129 222 [i.9].

**Table B.2.2-1: Mapping of MEC resources and CAPIF resources for service management**

MEC service management API		CAPIF APIs [i.9]	
resource name	resource URI	resource name	resource URI
Retrieve information about a list of mecService resources			
A list of mecService	mec_service_mgmt/v1/service	<i>CAPIF_Discover_Service_API</i> : All published service APIs	/service-apis/v1/allServiceApis
Retrieve information about a mecService resource			
Individual mecService	mec_service_mgmt/v1/services/{serviceId}	- (see note)	- (see note)
Retrieve information about the available transports			
A list of mecTransport	mec_service_mgmt/v1/transports	-	-
Retrieve information about a list of mecService resources of an application instance			
A list of mecService of an application instance	mec_service_mgmt/v1/applications/{appInstanceId}/services	<i>CAPIF_Publish_Service_API</i> : APF published APIs	/published-apis/v1/{apfId}/service-apis
Retrieve information about a mecService resource of an application instance			
Individual mecService of an application instance	mec_service_mgmt/v1/applications/{appInstanceId}/services/{serviceId}	<i>CAPIF_Publish_Service_API</i> : Individual APF published API	/published-apis/v1/{apfId}/service-apis/{serviceApfId}
Retrieve information about a list of mecSrvMgmtSubscription resources for this subscriber			
Parent resource of all mecSrvMgmtSubscription of a subscriber	mec_service_mgmt/v1/applications/{appInstanceId}/subscriptions	<i>CAPIF_Events_API</i> : CAPIF Events Subscriptions	/capif-events/v1/{subscriberId}/subscriptions/
Retrieve information about a mecSrvMgmtSubscription resource for this subscriber			
Individual mecSrvMgmtSubscription	mec_service_mgmt/v1/applications/{appInstanceId}/subscriptions/{subscriptionId}	<i>CAPIF_Events_API</i> : Individual CAPIF Events Subscription	/capif-events/v1/{subscriberId}/subscriptions/{subscriptionId}
NOTE: Although there is no resource defined in CAPIF for individual services, the query of a specific service is possible by using suitable filtering parameters with the CAPIF APIs.			

## B.2.3 Data models for service API discovery and publication

### B.2.3.1 Data model for services

The "ServiceInfo" data type, as defined in the present document, includes the following data types:

- TransportInfo
- SerializerTypes
- LocalityTypes
- SecurityInfo
- CategoryRef
- EndPointInfo

The "ServiceAPIDescription" is the corresponding CAPIF data type defined in ETSI TS 129 222 [i.9], which includes the following data types:

- AefProfile
- Version
- Resource
- CustomOperation
- Protocol
- DataFormat
- CommunicationType

- Operation
- InterfaceDescription
- ShareableInformation
- PublishedApiPath

Table B.2.3.1-1 shows a simplified version of data type definition for "ServiceAPIDescription". The full definition of all the related data types can be found in ETSI TS 129 222 [i.9].

**Table B.2.3.1-1: Definition of type ServiceAPIDescription [i.9]**

Attribute name	Data type	P	Cardinality	Description
apiName	string	M	1	API name.
apild	string	O	0..1	API identifier assigned by the CAPIF core function to the published service API.
aefProfiles	array(AefProfile)	C	1..N	AEF profile information, which includes the exposed API details (e.g. protocol).
description	string	O	0..1	Text description of the API
supportedFeatures	SupportedFeatures	O	0..1	The supported optional features of the CAPIF API.
shareableInfo	ShareableInformation	O	0..1	Represents whether the service API and/or the service API category can be published to other CCFs.
serviceAPICategory	string	C	0..1	The service API category to which the service API belongs to.
ccfld	string	C	0..1	CAPIF core function identifier which can be contacted further for discovering the details of service API information.
apiSuppFeats	SupportedFeatures	O	0..1	Provided by the consumer to indicate the features supported by the service API.
pubApiPath	PublishedApiPath	C	0..1	It contains the published API path within the same CAPIF provider domain.

### B.2.3.2 Data model for service API announcement/notification

Data types for MEC service availability subscriptions and notifications are defined in the present document as "SerAvailabilityNotificationSubscription" and "ServiceAvailabilityNotification".

The "EventSubscription" type in CAPIF, as defined in [i.9], includes the following types:

- CAPIFEvent
- CAPIFEventFilter

Table B.2.3.2-1 shows a simplified version of data type definition for "EventSubscription". The full definition of all the related data types can be found in [i.9].

**Table B.2.3.2-1: Definition of type EventSubscription [i.9]**

Attribute name	Data type	P	Cardinality	Description
events	array(CAPIF Event)	M	1..N	Subscribed events.
eventFilters	array(CAPIF EventFilter)	O	1..N	Subscribed event filters.
eventReq	ReportingInformation	O	0..1	Represents the reporting requirements of the event subscription.
notificationDestination	Uri	M	1	URI where the notification should be delivered to.
requestTestNotification	boolean	O	0..1	Set to true by Subscriber to request the CAPIF core function to send a test notification. Set to false or omitted otherwise.
websocketNotifConfig	WebsocketNotifConfig	O	0..1	Configuration parameters to set up notification delivery over Websocket protocol.
supportedFeatures	SupportedFeatures	O	0..1	Used to negotiate the supported optional features of the API.

The "EventNotification" type in CAPIF is defined in ETSI TS 129 222 [i.9] with the "CAPIFEventDetail" data type folded in. Table B.2.3.2-2 shows a simplified version of data type definition for "EventNotification". The full definition of all the related data types can be found in ETSI TS 129 222 [i.9].

**Table B.2.3.2-1: Definition of type EventNotification [i.9]**

Attribute name	Data type	P	Cardinality	Description
subscriptionId	string	M	1	Identifier of the subscription resource to which the notification is related - CAPIF resource identifier
events	CAPIFEvent	M	1	Notifications of individual events
eventDetail	CAPIFEventDetail	O	0..1	Detailed information for the event

**Draft**

## Annex C (informative): Analysis of EASProfile

Table C-1 shows the analysis of the EASProfile attributes, which is specified in Table 8.1.5.2.3-1 of 3GPP TS 29.558 [19].

**Table C-1: Analysis of the EASProfile attributes**

Attribute name	Data type	P	Cardinality	Description	Remarks
easId	string	M	1	The identifier of the EAS	No existing MEC attribute defined.
endPt	EndPoint	M	1	Endpoint information (URI, FQDN, IP address) used to communicate with the EAS. This information maybe discovered by EEC and exposed to ACs so that ACs can establish contact with the EAS.	<i>endPt</i> similar to <i>endpoint</i> defined for service producing applications (see note 1) in the present document and <i>EndPoint</i> similar to <i>EndPointInfo</i> defined in the present document.
aclds	array(string)	O	1..N	Identities of the Application Clients that can be served by the EAS	No existing MEC attribute defined.
provid	string	O	0..1	Identifier of the ASP that provides the EAS.	Similar to <i>appProvider</i> attribute in ETSI GS MEC 010-2 [4].
type	string	O	0..1	The category or type of EAS.	Similar to <i>CategoryRef</i> defined in the present document.
scheds	array(ScheduledCommunicationTime)	O	1..N	The availability schedule of the EAS.	No existing MEC attribute defined.
svcArea	ServiceArea	O	0..1	The list of geographical and topological areas that the EAS serves. ACs in the UE that are outside the area will not be served.	No existing MEC attribute defined.
svcKpi	EASServiceKPI	O	0..1	Service characteristics provided by the EAS.	Some attributes of <i>EASServiceKPI</i> similar to those of the <i>appCharcs</i> attribute within the <i>ApplicationList</i> defined in ETSI GS MEC 016 [20]. See note 2.
permLvl	array(string)	O	1..N	Level of service permissions supported by the EAS.	No existing MEC attribute defined.
easFeats	array(string)	O	1..N	Service features supported by the EAS.	No existing MEC attribute defined.
svcContSupp	array(ACRScenario)	O	1..N	The ACR scenarios supported by the EAS for service continuity. If this attribute is not present, then the EAS does not support service continuity.	No existing MEC attribute defined.
appLocs	array(RouteToLocation)	O	1..N	List of DNAI(s) and the corresponding N6 traffic routing information/routing profile ID, associated with the EAS.  It is a subset of the DNAI(s) associated with the EDN where the EAS resides.	No existing MEC attribute defined.
avlRep	DurationSec	O	0..1	The period indicating to the EES, how often the EES needs to check the EAS's availability after a successful registration.	No existing MEC attribute defined.

Attribute name	Data type	P	Cardinality	Description	Remarks
status	string	O	0..1	EAS status (e.g. Enabled, Disabled etc.)	Similar to <i>state</i> attribute defined in the present document.
NOTE 1: Currently only a service producing application would provide such information (and then only at runtime, since the endpoint is not part of the AppD since it is not an attribute of the TransportDescriptor).					
NOTE 2: The intent of the appCharcs is different, noting its description, e.g. "The application characteristics relate to the system resources consumed by the application" (i.e. MEC application, rather than AC). Whereas the IEs relating to the svcKpi are generally what can be offered to the AC by the EAS.					

Editor's note: Table C-1 may need to be updated in the cases EASProfile is updated by 3GPP.

Editor's note: Table C-1 may need to be updated, as it is FFS how MEC could introduce attributes similar to ones that are only currently defined in 3GPP TS 29.558 (referring to the EASProfile data structure).

Editor's note: Table C-1 may need to be updated in the cases where new attributes (e.g. *easId*) are subsequently introduced into MEC.

Editor's note: The data type definitions for the attributes *type*, *permLvl*, *easFeats* and *status* are FFS.

Editor's note: The inclusion of service APIs information in the EAS profile and its usage is FFS.

# Draft



## History

Document history		
V1.1.1	July 2017	Publication
V2.1.1	November 2019	Publication
V2.2.1	December 2020	Publication
V3.0.1	January 2021	Updated with the agreements in MEC(20)00424r1.
V3.0.2	November 2021	Updated with the agreements in MEC(21)00430r6.
V3.0.3	December 2021	Updated with the agreements in MEC(21)00586r1.
V3.0.4	December 2021	Clean-up done by <b>editHelp!</b> E-mail: <a href="mailto:edithelp@etsi.org">mailto:edithelp@etsi.org</a>
V3.0.5	January 2022	Updated with the agreements in MEC(22)00037r2, MEC(22)00051r1 and MEC(22)00052.

**Draft**